Volume 4, Issue 1, January 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Fields of File Records in the Root Directory for Data Hiding

Mrs.Shantala C.P
HOD,Dept of CSE,
Channabasaveshwara Institute of Technology,
Gubbi,Karnataka, India.

Mr.Visvanatha K.V
Professor,Dept,of CSE,
Channabasaveshwara Institute of Technology,
Gubbi,Karnataka,India

Abstract- Generally in steganographic techniques, we are hiding huge information. Suppose if we want to hide small information such as, password of a few characters then the use of special large size of cover media will result in loss of bandwidth. But if we hide the same small information in a file header which can hide 10 characters per file the problem of bandwidth can be solved. There are some fields in file header reserved for future use. These reserved bytes can be used to hide small information and changing these reserved bytes will not affect any file properties.

Keywords: FAT, JPEG steganography, sectors, root directory

I. INTRODUCTION

Steganography goal is to keep the presence of a message secret, or else even hiding t that communication is going on.. Whereas Cryptography [1] role is to obscure a message or communication so that it cannot be understood. Computer technology and the internet have made a breakthrough in the transmission of information. It has also opened a whole new way of applying steganography using computer technology. Secret information can be hidden in computer image files (jpeg, gif, bmp), audio files (wav, mp3), video files (mpeg, avi), or even text files. To the person who does not knowledgeable about the file containing hidden information, it looks perfectly normal. A stegoed image, even along with the Original image, if given to an average person, will be unable to tell any difference between the two, let alone deduce there is a message hidden in it. Provided the steganographic algorithm is good enough and the original image is not available, even an adept steganography expert would be unable to tell if an image contains hidden information. Making use of the Internet, secret information hidden in the carrier can be transmitted in quick session along with security and privacy. Image files are not the only carriers. Other files such as sound files (.wav, .mp3), video files (.mpeg), text files, any other directory structure or file can all be used to hide information. Information can also be hidden in intelligible text itself.

There are a large number of steganographic methods that most of us are familiar, ranging from invisible ink and microdots to secreting a hidden message in the second letter of each word of a large body of text and spread spectrum radio communication. By utilizing computers and its network, there are many other ways of hiding information, such as:

- Covert channels
- Hidden text within Web pages
- Hiding files in plain sight[2] `
- Null ciphers (e.g., using the first letter of each word to form a hidden message in an otherwise innocuous text)

Images are considered as the most popular file formats used in steganography. They are known for constituting a non-causal medium, due to the possibility to access any pixel[4] of the image in random fashion. In addition, the hidden information could remain invisible to the eye. Data in most of the steganographic systems seems to be embedded into the non-zero discrete cosine transform (DCT)[3] coefficients of JPEG images

II. LITERATURE SURVEY

Data can be hidden in digital media in a number of ways. The techniques can be broken down mainly into the following

- Insertion – based
- Substitution based
- Generation based

A. Insertion – based steganography

In this technique, data is inserted into another file. There are portions of a file those are ignored by the application. When data is inserted into these portions, it does not affect nor degrade the actual contents of the file; only the file size is increased. Some files have an EOF (end of file) marker as shown in the below figure. The application reading the file, when encounters this marker, steps reading any more data from the file, as it thinks it has reached the end of the file and there is no point trying to read any more.

The JPEG files are block oriented; there is a header for each JPG block. The well formed JPEG structure [5] file can be described by the following pseudo production rules as shown in Fig. 1.

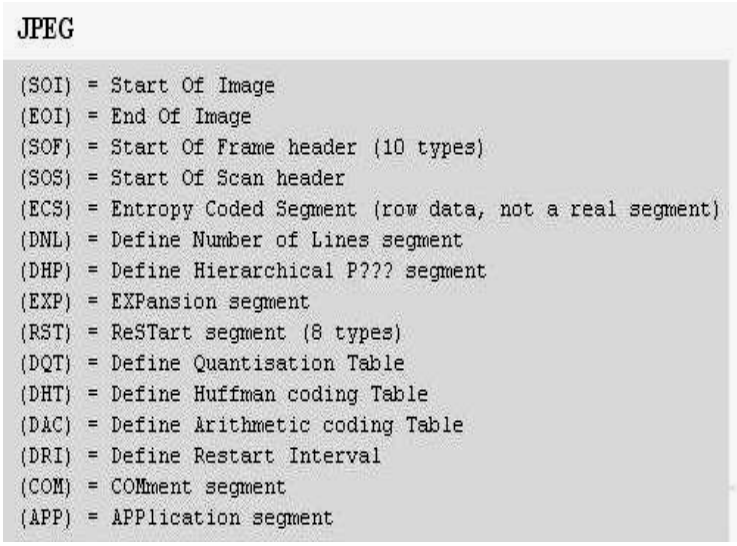

Fig.1.Structure of JPEG file.

This can be exploited to hide data. Data is simply inserted into the file after the EOF marker or EOI., and later can be easily retrieved. JPEG files have an EOF marker having the hex value FFD9. If data is inserted into the JEPG file after this marker, the file is not corrupted and behaved just like the original. The image in the file is also not even degraded. Only the file size is increased by the amount of data inserted into the image. An unlimited amount of data can be hidden this way. The insertion technique can also be used to hide data in text files. A tab character can be used to represent a binary 1, and a space character to represent a 0, these characters can be placed after the end of each line. Since these characters are invisible, unless the text application has options to show space and tab characters, they would fool anyone reading the text file. This type of steganography [6] is sometimes very easy to detect, and a file only attack is enough to break this type of steganography. The EOF technique can easily be detected; a hex editor can be used to open the file and see if there is any data after the EOF marker. This would require knowledge of the EOF marker for the particular file, as not all files use the same EOF value.

B. Substitution – based Steganography

This is most common form of steganography. It is a technique in which the bits of the file are replaced by bits of the data to be hidden as shown in the Fig. 2 . Only the bits of the file should be replaced which would have minimum impact on the file

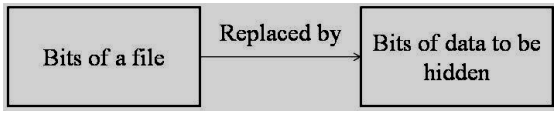

Fig 2.Substitution based steganography

The choice of the file is also an issue. It is not feasible to replace the bits in a text file; a single bit replaced would change the original character. Files those are suitable for ` type of steganography include image files, sound files, and video files. With this technique, there is a limit to the amount of data that can be hidden . Detecting this type of steganography depends on how and how many of the bits were replaced.

C. Generation – based Steganography

This form of steganography is quite different from the previous two. A carrier file is not required to hide the data in. The carrier will be in the form of signal, stream, or data file into which data is hidden by making subtle changes. Examples include audio files, image files, documents, or executable files. In practice, the carrier should look and work same as the original unmodified carrier, and should appear as benign to anyone inspecting it. Generation based stenography avoid using a carrier file.

The data to be hidden is itself used to generate a file. The data to be hidden is fed into an algorithm which generates a file based on that data. The generated file might not be perfect. For instance, a steganography program that takes as input a text file and generate a WAV file of classical music as output will unlikely produce something perfectly harmonic. This type of steganography is hard to detect as there is no original file which it can be compared with. However this type of steganography can best be detected with human perception than with anything else.

D. Problem Statement

Steganography became an important area of interest for individuals, corporations, and governments because of its unique technology of hiding data.. People are constantly communicating important information online due to the

increased connection caused by computers and the Internet. Steganography now is accomplished in the digital world.

Our approach is to hide the small information into the unused properties of a particular file. Precaution can be taken such that either the existing properties of the file cannot be changed or the actual contents of the file will be degraded. Only size of the file will be increased. Only the bits of the file should be replaced which would have minimum impact on the file

E. File Formats

The FAT file format [7] used by MS-DOS[8] and Windows 95 is a fairly simple file system in nature. Disks using the FAT file system are broken up into a fixed size chunks, called sectors. Consider for a floppy disk of 1.44MB, sectors are 512 bytes in size, which gives a total of 2880 sectors. The sectors of the disk are logically broken into 2 areas - the system sector, and the data sectors. The number of sectors that make up the system area is a nothing but function of the capacity of the disk. A 1.44 MB floppy disk is having 33 system sectors making up the system area and leaving 2847 data sectors for a total capacity of 1,457,664 bytes (1.44 MB).

System area: The system area is further broken down into 4 separate parts. Fig 3 shows the physical sector layout of system area of the disk and Fig 4 shows the parts of system area of disk.

Boot	FAT1	FAT2	Root directory
0	19	10.18	19.32

Fig 3:physical sector layout of disk

Boot sector
FAT
Copy of a FAT
Root directory

Fig 4:parts of system area

First part is the boot sector[11]. This sector contains the code that identifies the operating system that has been used to format the disk, and the "bootstrap" code that is necessary to start loading the old DOS operating system into main memory. The boot sector occupies sector 0 on the disk.

Second part in the system area is the File Allocation Table (FAT) which is used to keep track of the allocation of sectors to files, and does so by storing the next-sector information in the table. Each entry in the FAT is 12 bits in length. For example, if a file is located in sectors 5, 12, 35, and 96. An entry 5 in the FAT would hold the integer 12, entry 12 holds the integer 35 and entry 96 holds a special value indicating the end of the file as well. For a 1.44 MB floppy disk, the FAT takes up 9 sectors. Dealing with 12 bit entries in the FAT is more complex, since there is no data type in most programming languages that is 12 bits in length. In addition to regular sector numbers, an additional FAT entry might contain one of several special values.

For example:

- 0x000 indicates that this sector is currently not in use
- 0xFFF indicates that this in the last sector of a file

There is also a special value used to identity that the sector has been determined to be physically damaged, and should not be used. Third part in the system area is a copy of the FAT. It is used to make sure that if any part of the first FAT is corrupted, still the data on the disk can be accessed. Changes made to the FAT will reflect to both copies of the table.

Finally, the fourth part of the system area is the root directory [9]. There is one file record in the root directory for each file located in the root of that disk. Records are 32 bytes in length. The total number of sectors that the root directory occupies varies according to the capacity of the disk; for a 1.44 MB floppy, the root directory[10] occupies 14 sectors. Each file record in the root directory contains several fields as in below Table I.

TABLE I: FIELDS OF FILE RECORDS IN THE ROOT DIRECTORY AND THEIR SIZE IN BYTES

Filename:	8 characters (bytes)
Extension:	3 characters (bytes)
Attributes:	1 byte
Reserved:	10 bytes
Modtime:	2 bytes
Moddate:	2 bytes
Sstarting sector:	2 bytes
Size:	4 bytes

1) Filename: Contains the name of the file in ASCII text. There is a maximum of eight letters, all of which are stored in upper case. If the name is less than eight characters, the remainder of the field is filled with blanks. A zero in the first byte of the filename indicates that the directory entry has never been used. A value of 229 ($E5) in this byte indicates that the file has been erased.

2) Filename Extension: Contains three characters for the filename extension, stored in upper case. If the extension has

less than 3 characters, the remainder of the field is filled with blanks.

3) File Attribute: This field uses six of the bits in its byte to store certain file attributes.

4) Reserved: Currently this field is unused and reserved for future use. All bytes in this field are normally set to zero

5) Modtime: This field is treated as a 16-bit value divided into three sections. The high five bits are used for the hour (0-23). The middle six bits are used for the minutes (0-59). The low five bits contain seconds in two-second increments (0-29). Multiply the second's field by two to get the actual number of seconds.

6) Moddate: Also treated as a 16-bit value which is divided into three parts. The high seven bits are used for the year minus 1980 (add 1980 to the year value to get the correct year). The next four bits are used for the month The low five bits are used for the day of month (1-31).

7) First Cluster: Contains the number of the first cluster of the file. All following clusters in a file are found by tracing through the FAT, as explained above.

8) Starting-sector: It is one of the most important fields in the file record that specifies which of the data sectors contains the first sector of the file. It is important to know this fact such that we can use this sector to trace through the chain of sectors belonging to this file according to the next-sector information in the FAT.

9) File Size: A four-byte (or 32-bit) number containing the size of the file in bytes. The effective size of a file, when read by ST DOS, may be shorter than this if the last cluster is reached in the FAT before this many bytes have been read.

F. Attributes

As mentioned, the third field of the directory is one byte, of which six bits are used as file attribute flags. Currently, many of these attributes seem to be ignored by ST DOS, but the meanings are given here in case they get used by future releases of TOS.File Attributes are shown in Table II.

TABLE II
FILE ATTRIBUTES

Bit	Decimal Value	Hex Value	Meaning
0	1	1	Read Only
1	2	2	Hidden
2	4	4	System
3	8	8	Volume Label
4	16	10	Subdirectory
5	32	20	Archive
6	64	40	Unused
7	128	80	Unused

1) Read Only: When set, will prevent a file from being deleted or written to but will not prevent a file from being renamed, however.

2) Hidden and System: Have basically the same function. They should normally prevent the directory entry from appearing in a directory listing. These attributes are ignored by ST DOS and the Desktop.

3) Volume Label: Normally located only in the root directory of the directory structure. It marks a directory entry as containing the name of the volume or disk.

4) Subdirectory: Marks a directory entry as a subdirectory rather than a normal file. A subdirectory is the equivalent of a Desktop folder. Subdirectories are stored in the data space, just like files are. The structure of a subdirectory is the same as that of the main directory (usually called the root directory), except that a subdirectory is not fixed in size. A subdirectory only takes up as much space as it needs, but can grow to whatever size is necessary. There are two special entries in every directory which act as subdirectories themselves. They are "." and ".." and they refer to the current directory and the parent directory, respectively The entry called ".." in the root directory refers to the root directory itself.

5) Archive: Normally this field used only with hard disks. It should be set if a file has changed since it was last backed up.

6) Data area: Following the system area of the disk is the data area. It is place where the actual contents of files and subdirectories are stored. The sector numbers stored in the FAT actually refer to the offsets of the sectors relative to the very beginning of the data area. (For technical reasons, relative sectors 0 and 1 in the data area are reserved to be used in determining the length of the root directory. So the first relative sector number available to store data is data sector number 2, which is absolute sector 35.) The data sectors of any non-root directory are also composed of file records similar to what you see in the root directory. They provide information of file names, extensions, the starting data sectors, and so forth of the files in that directory. The data sectors of regular files will store the actual contents of the file.

III. METHODOLOGY

As we seen the third byte of a file is one byte from which last two bits are unused and also the fourth field which is of 10 byte is unused and is reserved for future use. We can use these bytes to hide our small secret information which will not affect the existing properties of the file or the directory structure. As the user opens the file the file attributes are not seen to the user.

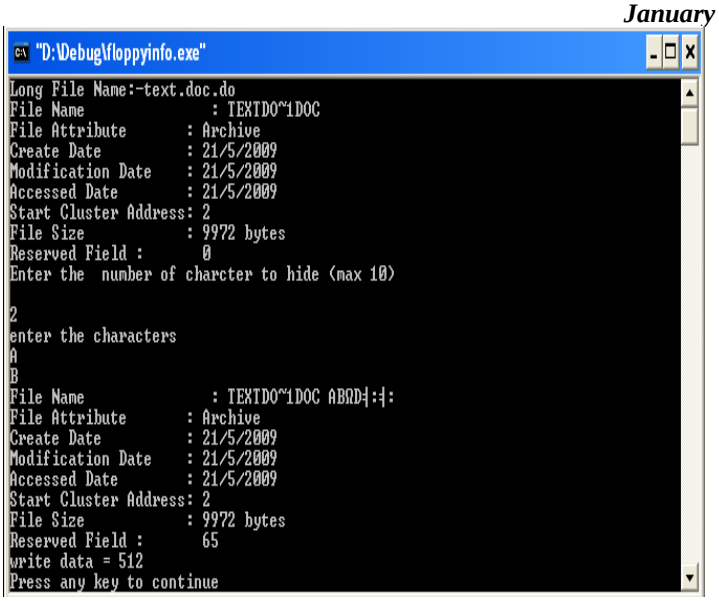

Fig 5: Hiding the two characters data into the file properties.

Fig 5. shows file properties and asks to enter the max number of characters to be hidden if it exceed the 10 characters (as 1 character is of 1 byte and we can hide maximum up to 10 bytes) then it shows a message telling you should not exceed the maximum character. In above example the number of character to hide is 2 then it asks the secret data to hide the file header here it is A and B and is hidden in a file named "text.doc".

Before giving the input, the reserved field was '0', after entering the input the reserved field will contain the ASCII value of the first character entered i.e., 'A'.

This shows that the characters entered is being written to a file named "text.doc", by displaying the value of reserved field as '65'. This is how we can hide 10 characters per file. Likewise we can create many such files and hide small information of 10 characters in each file.

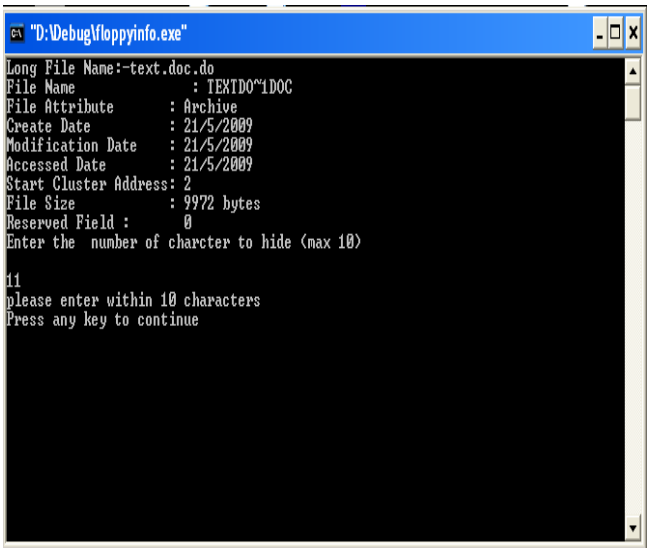

Fig 6: Maximum character to be hidden into the file header should not be exceeding 10 characters.

IV. ANALYSIS

When a JPEG image is modified as a result of steganography, certain colors will convert to another color according to the image color table. If a given color A occurs less frequently than B, A will be converted to B more often than B to A. Therefore the difference in color frequencies will decrease and an analysis of color frequency would not yield much information. Instead, an analysis of DCT coefficients should prove more fruitful.

V. CONCLUSION

We have explored the limits of steganographic theory and practice. Steganographic techniques can be used to hide data within digital media files with little or no visible change in the perceived appearance of that media file and can be exploited to export sensitive information. The proposes concept proves that securing a small information of few bytes can be achieved without effecting any extra overload of cover media file size. That is we can use the unused fields in the file

header to store the secured information and hence providing more secured and robust steganography scheme for smaller amount of information to be hidden into the cover medial files

REFERENCES

[1] Wayner P: Disappearing Cryptography, Information Hiding: Steganography and Watermarking (2nd edition), Morgan Kaufmann Publishers
[2] Cole E: Hiding in Plain Sight: Steganography and the Art of Covert Communication, Wiley Publishing, Inc.,
[3] Improved implementation of a modified Discrete Cosine Transform on low cost FPGA by S Belkouch
[4] A novel approach of image morphing based on pixel transformation by Jobayer Bin Bakkre & Rahman M T
[5] Jpeg Compression history estimation for Color images by Richardo de Queiroz and Richard Baranuik
[6] A new method in image steganography with improved image quality by Atallah M. Al-Shatnawi, Applied mathematical science Vol.6,2012,no.79,3907-3915.
[7] File format, From Wikipedia, the free encyclopedia: http://en.wikipedia.org/wiki/File_format
[8] A Description of the DOS File System - URL: http://alumnus.caltech.edu/~pje/dosfiles.html
[9] Root Directory and Regular Directories - URL: http://www.pcguide.com/ref/hdd/file/fatRoot-c.html
[10] FAT32 file system specification 2011 a white paper at staff.washington.edu/dittrich/misc/fatgen103.pdf
[11] FAT Boot Sector website – URL: http://averstak.tripod.com/fatdox/bootsec.htm

