SECURITY THREATS RELATED TO STEGANO IMAGES

PARIKSHIT MUNDA
Department of Computer Science& Engineering
C.I.T
Ranchi, India
sifuparixit@gmail.com

SUNITA GOLA
Department of Computer Science& Engineering
C.I.T
Ranchi, India
sunee.krish@gmail.com

BARKHA KUMARI
Department of Computer Science& Engineering
C.I.T
Ranchi, India
singhvarsha35@gmail.com

Abstract— **Steganography is the art and science of writing hidden messages in such a way that no one, apart from the sender and intended recipient, suspects the existence of the message, a form of security through obscurity. The advantage of steganography, over cryptography alone, is that messages do not attract attention to themselves, as they look like any normal bitmap or jpeg image. In this paper, we have discussed the security threat imposed by the steganography tools, and the proposed countermeasures. We have developed an algorithm which can compare a stegano image with normal images. Later we have discussed how this algorithm can be used to detect whether an image is a stegano image or not. We have used bitmap images in our research work.**

Keywords- steganography; stegano image; steganalysis; security; cover media

I. INTRODUCTION

Plainly visible encrypted messages, no matter how unbreakable, will arouse suspicion, and may in themselves be incriminating in countries where encryption is illegal.[1] Therefore, whereas cryptography protects the contents of a message, steganography can be said to protect both messages and communicating parties. Steganography includes the concealment of information within computer files. In digital steganography, electronic communications may include steganographic coding inside of a transport layer, such as a document file, image file, program or protocol. Media files are ideal for steganographic transmission because of their large size. As a simple example, a sender might start with an innocuous image file and adjust the color of every 100th pixel to correspond to a letter in the alphabet, a change so subtle that someone not specifically looking for it is unlikely to notice it. The focus of this paper is on alleged used of this technology of steganography by unfair bodies and the proposed countermeasures that an organization can implement having threat of illicit information being passed via its media channels.[9]

A. Alleged Use by Terrorists

When one considers that messages could be encrypted steganographically in e-mail messages, particularly e-mail spam, the notion of junk e-mail takes on a whole new light. Coupled with the "chaffing and winnowing" technique, a sender could get messages out and cover their tracks all at once.

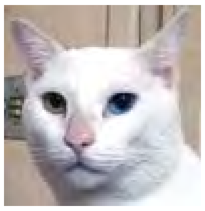


Figure 1.1

Figure 1.1 shows an example showing how terrorists may use forum avatars to send hidden messages. This avatar contains the message "Boss said that we should blow up the bridge at midnight." encrypted with http://mozaiq.org/encrypt using "växjö" as password.

Rumors about terrorists using steganography started first in the daily newspaper *USA Today* on February 5, 2001 in two articles titled "Terrorist instructions hidden online" and "Terror groups hide behind Web encryption". In July the same year, an article was titled even more precisely: "Militants wire Web with links to jihad". A citation from the article: "*Lately, al-Qaeda operatives have been sending hundreds of encrypted messages that have been hidden in files on digital photographs on the auction site eBay.com*". Other media worldwide cited these rumors many times, especially after the terrorist attack of 9/11, without ever showing proof. The Italian newspaper *Corriere della Sera* reported that an Al Qaeda cell which had been captured at the Via Quaranta mosque in Milan had pornographic images on their computers, and that these images had been used to hide secret messages (although no other Italian paper ever covered the story). The USA Today articles were written by veteran foreign correspondent Jack Kelley, who in 2004 was fired after allegations emerged that he had fabricated stories and sources.

In October 2001, the *New York Times* published an article claiming that al-Qaeda had used steganography to encode messages into images, and then transported these via e-mail and possibly via USENET to prepare and execute the September 11, 2001 Terrorist Attack. The Federal Plan for Cyber Security and Information Assurance Research and Development, [2] published in April 2006 makes the following statements:

- "…immediate concerns also include the use of cyberspace for covert communications, particularly by terrorists but also by foreign intelligence services; espionage against sensitive but poorly defended data in government and industry systems; subversion by insiders, including vendors and contractors; criminal activity, primarily involving fraud and theft of financial or identity information, by hackers and organized crime groups…" (p 9–10)
- "International interest in R&D for steganography technologies and their commercialization and application has exploded in recent years. These technologies pose a potential threat to national security. Because steganography secretly embeds additional, and nearly undetectable, information content in digital products, the potential for covert dissemination of malicious software, mobile code, or information is great." (p 41–42)
- "The threat posed by steganography has been documented in numerous intelligence reports." (p 42)

Moreover, an online "terrorist training manual", the "Technical Mujahid, a Training Manual for Jihadis" contained a section entitled "Covert Communications and Hiding Secrets Inside Images."[3]Despite this, *there are no known instances of terrorists using computer steganography*. Islamist use of steganography is somewhat simpler: In 2008 a British Muslim, Rangzieb Ahmed, was alleged to have a contact book with Al-Qaeda telephone numbers, written in invisible ink. He was convicted of terrorism.[4]

II. Research Basis

Today, there are a vast number of steganography tools available to create stegano images such as SecureEngine Professional [5]. Any person can avail these softwares and create stegano images. To break the security of these images, the following questions are required to be answered:

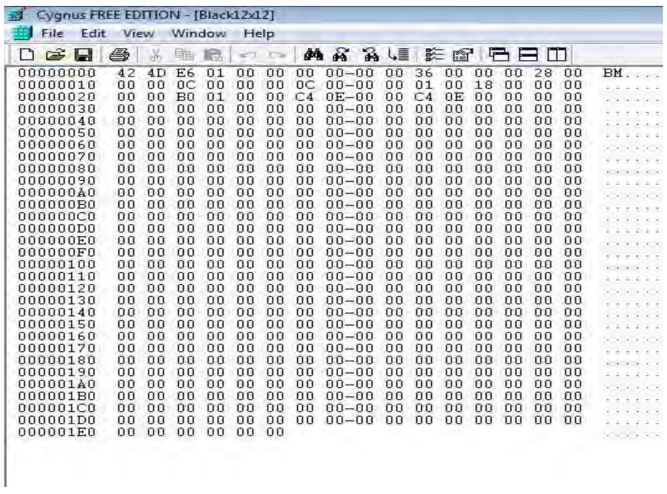
1. Whether the image is a stegano image?
2. In case of a stegano image, which algorithm is used for steganography?
3. Once the algorithm used for steganography is determined, what is the password given during steganography?

Let us assume that a person uses a simple 12x12 black image to encrypt a letter “a” using SecureEngine Professional [5] and uses AES algorithm and gives password “aaaa” during creating the stegano image.

As we know a related-key attack can break 256-bit AES with a complexity of 2^{119}, which is faster than brute force but is still infeasible. Hence recovering the encrypted password in this case is also infeasible. So we shall concentrate on finding a method which can easily be implemented to detect whether an image is a stegano image or not, so that the security measures can be taken later.

A. Initial Research Work

We have used bitmap images[6] in our research. It has been observed that when steganography is done on a black (all 0s except bmp header) 12x12 bitmap image then we have noticed change in its pixel value. Softwares used for this purpose are Hex Editor [7] and Secure Engine Pro v1.0.[5]

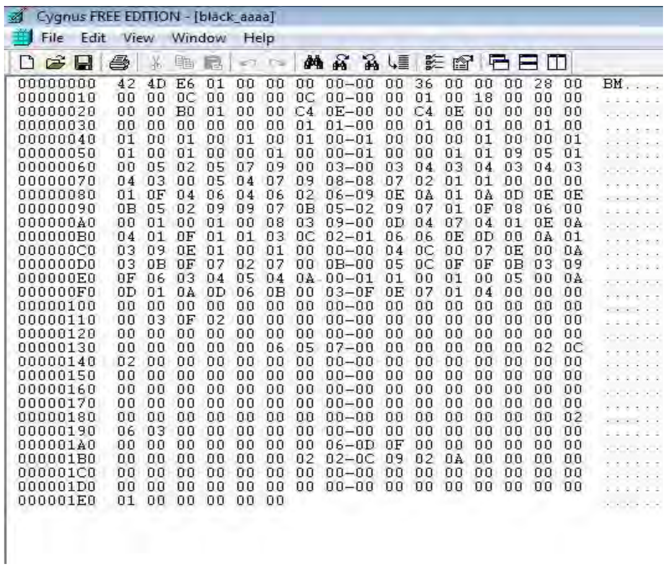

Cygnus FREE EDITION - [Black12x12]
File Edit View Window Help

```
00000000  42 4D E6 01 00 00 00 00-00 00 36 00 00 00 28 00   BM....
00000010  00 00 0C 00 00 00 0C 00-00 00 01 00 18 00 00 00   ......
00000020  00 00 B0 01 00 00 C4 0E-00 00 C4 0E 00 00 00 00   ......
00000030  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000040  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000050  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000060  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000070  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000080  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000090  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000000A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000000B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000000C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000000D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000000E0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000000F0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000110  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000130  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000140  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000180  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000190  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000001A0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000001B0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000001C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000001D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000001E0  00 00 00 00 00 00                                  ......
```

Figure 2.1

Figure 2.1 shows the pixel by pixel hex value of a black 12x12 bitmap image. This is to note that except the bmp header values, all pixel values for a complete black image is 00.

Using Secure Engine Pro v1.0.[5] we performed steganography on a black 12x12 bitmap image, the text file we have used contained only letter “a”, and the password used was “aaaa”.


Cygnus FREE EDITION - [black_aaaa]
File Edit View Window Help

```
00000000  42 4D E6 01 00 00 00 00-00 00 36 00 00 00 28 00   BM....
00000010  00 00 0C 00 00 00 0C 00-00 00 01 00 18 00 00 00   ......
00000020  00 00 B0 01 00 00 C4 0E-00 00 C4 0E 00 00 00 00   ......
00000030  00 00 00 00 00 00 01 01-00 00 01 00 01 00 01 00   ......
00000040  01 00 01 00 01 00 01 00-01 00 00 00 01 00 00 01   ......
00000050  01 00 01 00 00 01 00 00-01 00 00 01 01 09 05 01   ......
00000060  00 05 02 05 07 09 00 03-00 03 04 03 04 03 04 03   ......
00000070  04 03 00 05 04 07 09 08-08 07 02 01 01 00 00 00   ......
00000080  01 0F 04 06 04 06 02 06-09 0E 0A 01 0A 0D 0E 0E   ......
00000090  0B 05 02 09 09 07 0B 05-02 09 07 01 0F 08 06 00   ......
000000A0  00 01 00 01 00 08 03 09-00 0D 04 07 04 01 0E 0A   ......
000000B0  04 01 0F 01 01 03 0C 02-01 06 06 0E 0D 00 0A 01   ......
000000C0  03 09 0E 01 00 01 00 00-00 04 0C 00 07 0E 00 0A   ......
000000D0  03 0B 0F 07 02 07 00 0B-00 05 0C 0F 0F 0B 03 09   ......
000000E0  0F 06 03 04 05 04 0A 00-01 01 00 01 00 05 00 0A   ......
000000F0  0D 01 0A 0D 06 0B 00 03-0F 0E 07 01 04 00 00 00   ......
00000100  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000110  00 03 0F 02 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000120  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000130  00 00 00 00 00 06 05 07-00 00 00 00 00 00 02 0C   ......
00000140  02 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000150  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000160  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000170  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
00000180  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 02   ......
00000190  06 03 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000001A0  00 00 00 00 00 00 00 06-0D 0F 00 00 00 00 00 00   ......
000001B0  00 00 00 00 00 00 02 02-0C 09 02 0A 00 00 00 00   ......
000001C0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000001D0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ......
000001E0  01 00 00 00 00 00                                  ......
```

Figure 2.2

Figure 2.2 shows the pixel by pixel hex value of the same black image after steganography is performed on it. This is to note that some pixel values have noticeably changed to some specific values. However it is very difficult to observe any change in the color of the stegano image as normally in steganography the least significant bits (LSBs) are used to store bit values of the encrypted text file and the password.

The research work is following such changes for different bitmap images, text files, and password values. So that we could be finally supposed to be able to determine that in a stegano image, which pixels are generally used to store the bit values of the encrypted text file and the password.

B. Complexities

Initially we have thought about finding out which pixels are actually used in a bitmap image to store the bit values of a text file in the bitmap image. We have found after multiple research that each time a steganography is performed on a bitmap image using same text file and the password, even then every time changes occurs in different pixel values without any fixed pattern detected in the change of pixel positions used for hiding the bit values.

Our research shows that on a same black 12x12 bitmap image when steganography is performed on different time instances using a same text file containing only letter ”a” and using the same password ”aaaa”, then every time changes have been noticed in different pixel positions. Hence we arrived at a conclusion that these changes in the pixel positions actually depends on the system time. In our next research we first froze the system time and performed the above research again. Disappointingly we noticed that still the pixel positions used during steganography are not constant. Satisfied though, we found that the changes occurring now are quite less than when the system time was not frozen.

Hence now we came to a conclusion that these changes in the pixel positions surely depend on the system time and some other factors which are still not detected.

C. Conclusion to our Initial Research Work

Hence we concluded that at least at this point of time with available piece of knowledge it is vague trying to find out the pixel positions used during a steganography process. So we shifted to some different logics to sort out our problems. Our attempt is to create an algorithm, using which we can conclude that an image may be a stegano image so that the necessary security measures can be followed.

III. Work Performed

We have developed an algorithm which can take 8-bit binary input in series from a stegano image and can compare it to some other normal bitmap images. We have arrived to some conclusions after this attempt, which is discussed in detail later in this paper. Given below is the basic algorithm used in our research, implemented in C language. Several modifications can be done on this algorithm as per the requirement of the research work. E.g. following algorithm takes 1 stegano image and 4 normal images as input. The number of normal images to be compared with the stegano images can be changed as per the requisite of the research work.

A. 3.1 Algorithm to compare a stegano image with normal images

```
Input:
File1(Stagano image)
File2(image1)
File3( image2)
File4( image3)
File5( image4)
Declare:
Var_unique_code
Var_f1
Var_matched = false
Var_i
Start:
Open file pointer F1,F2,F3,F4 and F5(in binary mode)
F1 points to File 1
…
…
…
F5..to File F5
Loop l1
Var_i=0
Read the 8-bit binary code pointed by F1  and store in Var_f1
F1 is incremented to point the next 8-bit binary code.
i++
Loop l2
Var_i=0
If Var_matched = =true, then exit
Compare var_f1 with the first 8-bit binary code in the file pointed by F2.
F2 is incremented to point the next 8-bit binary code.
If a match is found then Var_matched = true and exit l2
Else if  end of file 8-bit binary code is found then exit l2
End loop l2
```

```
Loop l3
If Var_matched = =true, then exit
Compare var_f1 with the first 8-bit binary code in the file pointed by F3.
F3 is incremented to point the next 8-bit binary code.
If a match is found then Var_matched = true and exit l3
Else if end of file 8-bit binary code is found then exit l3
End loop l3
Loop l4
If Var_matched = =true, then exit
Compare var_f1 with the first 8-bit binary code in the file pointed by F4.
F4 is incremented to point the next 8-bit binary code.
If a match is found then Var_matched = true and exit l4
Else if  end of file 8-bit binary code is found then exit l4
End loop l4
Loop l5
If Var_matched = =true, then exit
Compare var_f1 with the first 8-bit binary code in the file pointed by F5.
F3 is incremented to point the next 8-bit binary code.
If a match is found then Var_matched = true and exit l5
Else if  end of file 8-bit binary code is found then exit l5
End loop l5
If Var_matched = false, then
print Var_unique_code in decimal
print Var_i
Var_matched = false
End loop l1
```

Remark: This algorithm will generate all the unique 8bit codes in decimal present in the stegano image relative to the normal image and also the position of the 8bit code using the counter "i".

B. Some results to be discussed after a C code implementation of the algorithm (3.1)

Case 1:

We took one stegano image and compared with 4 other normal images, one of them being the original image of the stegano image.

Output: Some non-unique and some unique code are generated.

Remarks: Proves that all unique codes found in the stegano image contains the ascii values in them encrypted.

Case 2:

Same as case 1 but the stegano image to be compared is replaced with its original image. Hence same original image is compared with itself.

Output: No unique code is generated.

Remarks: Our program is working perfect.

Case 3:

The stegano image is compared with its own original image only which was used during steganography.

Output: Some non-unique and some unique code are generated. The number of unique codes in this case is more than that in case 1.

Remarks: Only these binary codes are used to encrypt the ascii values of the text file and password encrypted.

Case 4:

The stegano image is compared with 4 other normal images, none of them being the original image of the stegano image.

Output: Some non-unique and some unique code are generated.

Case 5:

The stegano image is compared with only one normal image, not being the original image of the stegano image.

Output: More unique codes are generated this time than in case 4.

Case 6:

The original image of the stegano image is compared with same set images used to compare the stegano image in Case 4.

Output: Exactly same number of unique codes generated as in Case 4.

Remarks: Most of the 8-bit binary codes in an image are altered after performing a steganography upon it. But as these 8-bit binary codes are actually the pixel intensity of red, blue and green colors in an image, they actually define that image. After Steganography these intensities are altered minutely, as LSBs are used for encryption. Hence even after steganography, though difference in codes is found between the stegano and the original image, the difference in codes of the stegano image with any other image and that of its original image is exactly the same in most cases.

IV. Proposed Work

As we have discussed that except the bmp header values, all pixel values for a complete black image is 00. Similarly in a complete white image, all pixel values except the header file are FF. All uniform colored images generally follow a pattern as given below:

Red: 24 1C ED 24 1C ED 24 1C ED

Blue: F3 6D 4D F3 6D 4D F3 6D 4D

Green: 4C BL 22 4C BL 22 4C BL 22

Similarly all uniformed colored bitmap images follows a uniform pattern. One can find these pattern by creating any uniform colored bitmap image using Microsoft paint software[8] and then opening them using Hex editor[7]. This pattern gets altered when the image goes under steganography.

Now let us assume a stenographer has used the following image (Figure 4.1) as carrier to hide a text file.

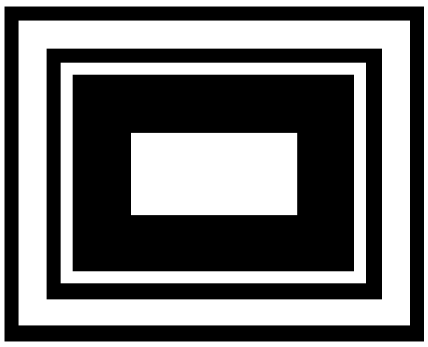


Figure 4.1

Given below is the algorithm to determine whether this image is a stegano image or not.

A. 4.1 Algorithm to detect a stegano image

Step1: Crop a portion of the image which is of uniform color or is likely to have fixed pattern when opened with hex editor (Discussed above in 4).

Step 2: Create another image using Hex editor[7] having the same pattern. E.g. in this image we will crop a complete black portion of the image and will create another image having all hex values 00 after the bmp header values.

Step 3: Compare the two images using Algorithm 3.1.

Step 4: If some unique codes are found, then print “this image may be a stegano image”.

Remarks1: Pixel positions of these unique codes are supposed to contain some of the bit values of the hidden text file.

Remarks2: In some cases, this algorithm may detect a normal image as stegano image but never a stegano as normal image. Through more research work, we can arrive to a conclusion that upto what percentage of unmatched codes can be allowed to certify an image as normal image and embed the logic in Step4.

Note: Though the example of a black & white image is given here, the algorithm works fine with any image once the hex value pattern is determined correctly.

V. CONCLUSION

As discussed above, to avoid such alleged use of the technology of steganography, the organizations responsible for data channels should embed some technology as discussed in this paper. Using such technology, an organization can detect if any person is using its data channels to send or receive covered media files. A person, who is not authorized to send stegano images or encrypted files through its data channels, must be debarred from doing so. Any file arising suspect can be withheld and the sender can be informed about it and some clarifications can be demanded.

Note: The Algorithms prescribed in this paper are successful in our research work only and the usefulness of these algorithms in real time systems and in the industry is yet to be tested. We, the authors, do not emphasize to implement our algorithm as it is, as still a vast amount of research is to be done in this topic.

REFERENCES

[1] Pahati, OJ (2001-11-29). "Confounding Carnivore: How to Protect Your Online Privacy". AlterNet. Archived from the original on 2007-07-16. http://web.archive.org/web/20070716093719/ http://www.alternet.org/story/11986/. Retrieved 2008-09-02.

[2] CSIA12i-FINAL.qxd

[3] The Jamestown Foundation

[4] http://www.dailymail.co.uk/news/article-1061190/British-Muslim-Al-Qaeda-contacts-book-terrorists-numbers-written-invisible-ink.html

[5] securengine.apponic.com

[6] $^{a\ b\ c}$ MSDN - BITMAPINFOHEADER: BI_ALPHABITFIELDS in biCompression member

[7] www.hexworkshop.com/ microsoft.com/resources

[8] http://www. /documentation/windows/xp/all/proddocs/en-us/mspaint_overview.mspx?mfr=true

[9] Introduction part - www.wikipedia.com