Data hiding Algorithm for Bitmap Images using Steganography

Mamta Juneja

Department of computer science and Engineering,RBIEBT,Sahuran
er_mamta@yahoo.com

Abstract: Steganography is related to cryptography and is the basis for many of the digital watermarking techniques currently being developed. The interest in data hiding has risen with the recent activity in digital copyright protection schemes. As the information age progresses and data become more and more valuable, methods need to be discovered to protect and secure sensitive data. This paper will discuss the applications of steganography including hiding data within: text, mpeg layer three and images. This paper also covers basic algorithm for implementing steganography within bitmap images.

[Mamta Juneja. **Data hiding Algorithm for Bitmap Images using Steganography**. Researcher. 2010;2(12):67-73]. (ISSN: 1553-9865). http://www.sciencepub.net.

Keywords: Steganography, Bitmap images, Cryptography.

1. Introduction:

Cryptography is simply the science of protecting data so that nobody reads it that isn't authorized to do so. It allows an individual to encode data so the recipient will be the only person able to decipher the data. Steganography is in many ways an obscure type of cryptography. It allows a person to hide data within other data with the intent of not drawing suspicion to the context in which the message was transferred. Steganography can be used to take a document and hide that document's data within an image. This method of cryptography has a variety of applications that all result in the combination of two different documents into one innocuous document. This resulting data hides the sensitive information inside the other data in such a way that it shouldn't be noticeable.

Steganography is more popularly known as "Digital Watermarking". Unlike its parent cryptography, digital watermarking doesn't depend on a symmetric or asymmetric encryption scheme. For instance, in an asymmetric encryption scheme the algorithm relies on a public/private key pair to encrypt and decrypt a message and in a symmetric algorithm the message is encrypted and decrypted with the same key. Steganography uses neither of these methods and relies on data hiding to accomplish its task. The aim of it is to obscure the data so much that no one would ever think to extract data from the target. It gained the name digital watermarking because this method can be used to store copyrights and trademarks within data to protect the owner of the document. Such organizations as the RIAA and MPAA are researching this method of information hiding to imbed special tags to make it easier to catch and prosecute music and movie pirates .

Steganography can also be used to verify data integrity. For example, take a document or image and encode a few words such as "Hello world" over the entire document's length. One can then ensure data integrity by reversing the steganography process to see if the "Hello World" comes back as the hidden data. If the data is mangled that means that the original file has been changed. This data verification is just a branch of the watermarking concept and similar methods are used to watermark copyrighted information.

The technology as a whole isn't perfect. The data is only secured because it isn't apparent that "hidden data" even exists. The more data that is hidden within a document the more the document changes. For instance, an image may become more distorted and artifacts may occur, a document's formatting or content may change, and an mp3 may have additional static or white noise. Popular tools exist that detect this change in a document; the process is known as steganalysis. This procedure is performed on a piece of data to determine the likelihood of hidden data existing within document. If the conclusion is made that data does in fact reside within the document, it is only a matter of time before the data is discovered.

1.1 Digital Steganography

In a typical digital steganographic encoder, message is the data that the sender wishes to remain confidential and can be text, images, audio, video, or any other data that can be represented by a stream of bits. The cover or host is the medium in which the message is embedded and serves to hide the presence of the message. This is also referred to as the message wrapper. The message embedding technique is strongly dependent on the structure of the cover, and in this paper covers are restricted to being digital images. It is not required that the cover and the message have homogeneous structure. For example, it is possible to embed a recording of Shakespeare's lines (an audio stream message) inside a digital portrait of the famous playwright (an image cover).

The image with the secretly embedded message produced by the encoder is the stego-image. The stego-image should resemble the cover image under casual inspection and analysis. In addition, the encoder usually employs a stego-key which ensures that only recipients who know the corresponding decoding key will be able to extract the message from a stego-image.

Recovering the message from a stego-image requires the stego-image itself and a corresponding decoding key if a stego-key was used during the encoding process. The original cover image may or may not be required; in most applications it is desirable that the cover image not be needed to extract the message.

Applications

There are many applications for digital steganography of images, including copyright protection, feature tagging, and secret communications [1,4].

Copyright Protection: A secret copyright notice or watermark can be embedded inside an image to identify it as intellectual property [5, 6]. This is the watermarking scenario where the message is the watermark [5, 6]. The "watermark" can be a relatively complicated structure. In addition, when an image is sold or distributed an identification of the recipient and time stamp can be embedded to identify potential pirates. A watermark can also serve to detect whether the image has been subsequently modified [7]. Detection of an embedded watermark is performed by a statistical, correlation, or similarity test, or by measuring other quantity characteristic to the watermark in a stego-image. The insertion and analysis of watermarks to protect copyrighted material is responsible for the recent surge of interest in digital steganography and data embedding.

Feature Tagging: Captions, annotations, time stamps, and other descriptive elements can be embedded inside an image, such as the names of individuals in a photo or locations in a map. Copying the stego-image also copies all of the embedded features and only parties who possess the decoding stego-key will be able to extract and view the features. In an image database, keywords can be embedded to facilitate search engines. If the image is a frame of a video sequence, timing markers can be embedded in the image for synchronization with audio. The number of times an image has been viewed can be embedded for "pay-per-view" applications.

Secret Communications: In many situations, transmitting a cryptographic message draws unwanted attention. The use of cryptographic technology may be restricted or forbidden by law. However, the use steganography does not advertise covert communication and therefore avoids scrutiny of the sender, message, and recipient. A trade secret, blueprint, or other sensitive information can be transmitted without alerting potential attackers or eavesdroppers.

Features of Steganograpic Techniques

Steganographic techniques embed a message inside a cover, various features characterize the strengths and weaknesses of the methods. The relative importance of each feature depends on the application [4].

Hiding Capacity: Hiding capacity is the size of information that can be hidden relative to the size of the cover. A larger hiding capacity allows the use of a smaller cover for a message of fixed size, and thus decreases the bandwidth required to transmit the stego-image.

Perceptual Transparency: The act of hiding the message in the cover necessitates some noise modulation or

Distortion of the cover image: It is important that the embedding occur without significant degradation or loss of perceptual quality of the cover. In a secret communications application, if an attacker notices some distortion that arouses suspicion of the presence of hidden data in a stego-image, the steganographic encoding has failed even if the attacker is unable to extract the message. Preserving perceptual transparency in an embedded watermark for copyright protection is also of paramount importance because the integrity of the original work must be maintained [6].

For applications where the perceptual transparency of embedded data is not critical, allowing more distortion in the stego-image can increase hiding capacity, robustness, or both.

Robustness: Robustness refers to the ability of embedded data to remain intact if the stego-image undergoes transformations, such as linear and non-linear filtering, addition of random noise, sharpening or blurring, scaling and rotations, cropping or decimation, lossy compression, and conversion from digital to analog form and then re-conversion back to digital form (such as in the case when a hard copy of a stego-image is printed and then a digital image is formed by subsequently scanning the hardcopy.) Robustness is critical in copyright protection watermarks because pirates will attempt to filter and destroy any watermarks embedded in images [5, 6]. Anti-watermarking software is already available on the Internet and have been shown effective in removing some watermarks [8,9]. These techniques can also be used to destroy the message in a stego-image.

Tamper Resistance: Beyond robustness to destruction, tamper-resistance refers to the difficulty for an attacker to alter or forge a message once it has been embedded in a stego-image, such as a pirate replacing a copyright mark with one claiming legal ownership. Applications that demand high robustness usually also demand a strong degree of tamper resistance. In a copyright protection application, achieving good tamper resistance can be difficult because a copyright is effective for many years and a watermark must remain resistant to tampering even when a pirate attempts to modify it using computing technology decades in the future.

Other Characteristics: Computational complexity of encoding and decoding is another consideration and individual applications may have additional requirements. For example, for a copyright protection application, a watermark should be resistant to collusion attacks where many pirates work together to identify and destroy the mark.

1.2 Data Embedding

Current methods for the embedding of messages into image covers fall into three categories: Least-Significant Bit embedding (or simple embedding), transform techniques, and methods that employ perceptual masking.

1.2.1 Least-Significant Bit Encoding

A digital image consists of a matrix of color and intensity values. In a typical gray scale image, 8 bits/pixel are used. In a typical full-color image, there are 24 bits/pixel, 8 bits assigned to each color components.

The simplest steganographic techniques embed the bits of the message directly into the least-significant bit plane of the cover image in a deterministic sequence. Modulating the least-significant bit does not result in a human-perceptible difference because the amplitude of the change is small. Other techniques “process” the message with a pseudo-random noise sequence before or during insertion into the cover image.

The advantage of LSB embedding is its simplicity and many techniques use these methods [10]. LSB embedding also allows high perceptual transparency. However, there are many weaknesses when robustness, tamper resistance, and other security issues are considered. LSB encoding is extremely sensitive to any kind of filtering or manipulation of the stego-image. Scaling, rotation, cropping, addition of noise, or lossy compression to the stego-image is very likely to destroy the message. Furthermore an attacker can easily remove the message by removing (zeroing) the entire LSB plane with very little change in the perceptual quality of the modified stego-image.

1.3 Implementation:

The algorithm developed is a basic implementation of steganography. There is no “right” way to implement steganography because if there were it would be very easy to run thousands of images through the reverse steganography process to extract the possible data. So in all actuality the more random an implementation of steganography is, the harder it will be to extract the data. The Algorithm

The following algorithm is implemented in pseudo code with comments. The actual implementation was written in C++. This code works for bitmap images (or any image that has lossless compression). Compression ruins the watermarking for obvious reasons; it destroys the small changes that the algorithm makes to each pixel. The actual loading and writing of image data is library and platform specific and is beyond the scope of the paper. This code is only for the encoding portion of the data, to decode, these steps must be taken in reverse order.

```
integer height, width;
2DArray imgbuffer = LoadImage("imagefile", &height, &width); //load data of our image into
matrix

string txtbuffer = LoadText("textfile"); //load the data to be hidden

stringofbits binarydata; // the number of bits will be 8 * length(txtbuffer)

integer index;
integer x, y;
integer Red, Green, Blue;

for[index = 0; index < length(txtbuffer); index++]

        binarydata += GBDFA ( txtbuffer[index] ); //get the binary data from the ascii character

end for

index = 0;

//now iterate through each pixel of data encoding 3 bits into every image by changing the least
//significant bit of each color value

for[x = 0; x < width; x++]

        for[y = 0; y < height; y++]

                GetRGB(&Red, &Green, &Blue, imagebuffer[x][y]); /get the current values

                //encode the bit of the data into the color
                EncodeBitIntoColor(binarydata[i++],  Red); //fails if index doesn't exist
                if i == length(binarydata)
                        breakout of loop;

                EncodeBitIntoColor(binarydata[i++],Green);
                if i == length(binarydata)
                        breakout of loop;

                EncodeBitIntoColor(binarydata[i++],Blue);
                if i == length(binarydata)
                        breakout of loop;

                SetRGB(Red,Green,Blue,imagebuffer[x][y]); //write new colors to the pixel data


        end for

end for

SaveImage("imagefile"); //save the new image
```

This code reads two files from disk: the document that needs to be hidden and the image that will serve as the hiding place. The document that was read from disk is then converted into the 8 bit binary equivalents each character's ASCII value. For example: the letter A is found in the document; that is 97 ASCII which is 01100001 binary. Once all of the characters in the document are converted to binary, the image's color values are read into a matrix that corresponds to each pixel in the image. The algorithm modifies the least significant bit of each pixel containing red, green and blue values. Each pixel in the image matrix is modified until all of the document data is hidden.

For example:

Pixel 0,0 contains the color values 255,255,255 which are 11111111, 11111111, 11111111 in binary (a very bright white).

The data we are encoding is the first three bits of the capital letter A: 011.

The encoding would make the values change in this order
11111111 becomes 11111110
11111111 stays as 11111111
11111111 stays as 11111111

and the resulting color triple is 254,255,255 which is a shade of white that has an indiscernible difference compared to the previous color. What this means is that the Steg-Image will look nearly identical to the original image.

Once all this data is encoded into the image data in memory it is written back to disk with the hidden data.

This algorithm does not add a tag to show the reverse steganography algorithm where to stop and start. This would be necessary so that junk data is not extracted from the file.
This algorithm does not verify the size of the document data and the image data. The max character data that can be stored in the image is:
((Length * Width * 3)/8) - (SIF).

Where Length and Width are in number of Pixels and SIF is the length in characters of the signature files appended to the data that needs to be hidden. If this is not checked errors can and will occur in the encoding and decoding of data.

The things mentioned here were not implemented for one of two reasons. The first reason being the fact that some of the image saving and rewriting is far too operating system/library dependent to be included in the pseudo-code; meaning it would be cumbersome and unnecessary to include this code. Secondly some of the code was excluded because it was not the core part of the algorithm and the actual implementation of the code left out could easily be implemented in a variety of ways.

Results:

The cover image Lena is shown on Figure 2. The message is a text file containing a single line: “Hello World.” The stego-image shown on Figure 3 is produced. The difference image is shown on Figure 4, where “white” pixels indicate the spatial locations where the images differ.

Suggestions for improvement

Compress, encrypt and then hide data. This adds two layers of protection. The file name could provide the decryption key. While this method may not be perfect, it provides an additional layer of obscurity.
Use the last two bits of every color value resulting in greater image distortion but also a greater amount of data stored in the image. This would allow 6 bits to be stored per pixel instead of three.
When implementing the signature information make it variable so that someone cannot randomly look through images for hidden data. Only the persons that watermarked the image would have a clue where the image data stopped and started.

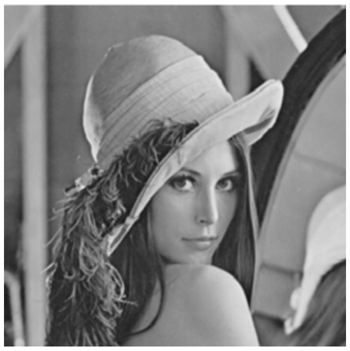
Figure 2: Cover Image

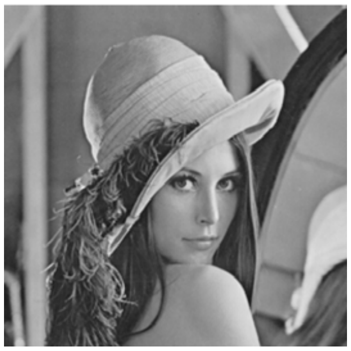
Figure 3: Steganos Stego-Image

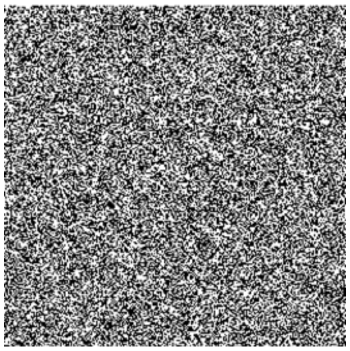
Figure 4: Difference Image

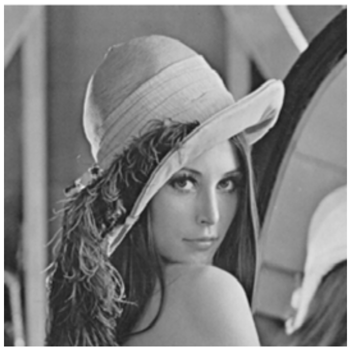
Figure 5: Modified Stego-Image

1.4 Conclusions

Steganography is an effective way to obscure data and hide sensitive information. It allows an individual to hide data inside other data with hopes that the transfer medium will be so obscure that no one would ever think to examine the contents of the file. Using the basic pseudo-code algorithm described above, it is possible to implement a steganography program to do all of the things described in this document: hide sensitive data, watermark images, and verify data integrity.

Although steganography is presented as a way to hide and watermark data is should be noted that the technology is not perfect. This method just secured data by obscuring the context in which it was transferred. If a prospective attacker wants to obtain the hidden, sensitive information this is just another barrier to discourage discovery. Like anything relating to security, if the attacker wants the data badly enough it will be obtained. For the time being, steganography seems to be a technology on the frontier of cryptographic methods that is just in its early stages. It serves as a new way to hide sensitive data but the overall "guessing game" that one plays with securing data makes steganography less appealing than other cryptographic methods. With continued research and an improvement in algorithms design, steganography can be taken as a serious way to hide data.

References

1. N. Johnson and S. Jajodia, "Exploring steganography: seeing the unseen," IEEE Computer, pp. 26-34, February 1998.

2. D. Kahn, The Codebreakers, Macmillian, New York, 1967.

3. B. Norman, Secret Warfare, Acropolis Books, Washington D.C., 1973.

4. W Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for data hiding," IBM Systems Journal, Vol. 35, No. 3 and 4, pp. 313-336, 1996.

5. M. Swanson, M. Kobayashi, and A. Tewfik, "Multimedia data embedding and watermarking technologies," Proceedings of the IEEE, Vol. 86, No. 6, pp. 1064-1087, June 1998.

6. R. Wolfgang, C. Podilchuk and E. Delp, “Perceptual watermarks for images and video,” to appear in the Proceedings of the IEEE, May, 1999. (A copy of this paper is availavle at: http://www.ece.purdue.edu/~ace).

7. R. B. Wolfgang and E. J. Delp, “Fragile watermarking using the VW2D watermark,” Proceedings of the SPIE/IS&T Conference on Security and Watermarking of Multimedia Contents, SPIE Vol. 3657, San Jose, CA, January 1999.

8. UnZign software: http://altern.org/watermark, 1997.

9. Stirmark software: http://www.cl.cam.ac.uk/~fapp2/watermarking/stirmark, 1997.

10. N. Johnson and S. Jajodia, “Steganalysis of images created using current steganography software,” Lecture Notes in Computer Science, Vol. 1525, pp. 273-289, 1998.

11. Steganos Software: http://www.demcom.com/english/steganos/index.htm

12. I. Cox, J. Kilian, T. Leighton, and T. Shamoon, “Secure spread spectrum watermarking for multimedia,” IEEE

Transactions On Image Processing, Vol. 6, No. 12, pp. 1673-1687, December 1997.

13. I. Cox and M. Miller, “A review of watermarking and the importance of perceptual modeling,” Proceedings of the

SPIE/IST&T Conference on Human Vision and Electronic

Imaging II , SPIE Vol. 3016, San Jose, CA, pp. 92-99, February 1997.

14. M. Swanson, B. Zhu, and A. Tewfik, “Robust data hiding for mages,” Proceedings of the IEEE DSP Workshop, Leon, Norway, pp. 37-40, Loen, Norway, September 1996.

15. R. Anderson and F. Petitcolas, “On the limits of steganography,” IEEE Journal on Special Areas in Communications, Vol. 16, No. 4, pp. 463-473, May 1998.

16. J. Smith and B. Comiskey, “Modulation and information hiding in images,” Lecture Notes in Computer Science, Vol. 1174, pp. 207-226, 1996.

17. F. Petitcolas and R. Anderson, “Weaknesses of copyright marking systems,” Proceedings of the ACM Multimedia and Security Workshop (at ACM Multimedia ’98), pp. 55-62, Bristol, United Kingdom, September 1998.

18. Introduction to Cryptography - http://www.pericosecurity.com/products/ikey/crypto.htm

19. Basic Steganography Concepts - http://www.stegoarchive.com/ http://www.techtv.com/audiofile/features/story/0, 23008,3305128,00.html.

8/4/2010