Using image steganography for decryptor distribution

T. Morkel [1], J.H.P. Eloff [2], M.S. Olivier [3]

Information and Computer Security Architecture (ICSA) Research Group
Department of Computer Science, University of Pretoria, 0002, Pretoria, South Africa
{tmorkel, eloff, molivier}@cs.up.ac.za

Abstract. When communicating secret information there is more than one route to follow to ensure the confidentiality of the message being transmitted. Encryption might be an obvious choice; however there are limitations and disadvantages to using encryption. An alternative approach is steganography, which is a technology for hiding information in other information. Combining the two disciplines may provide better security but more overhead, since the receiver must now have knowledge not only of how the information was encrypted, but also of how it was hidden, in order to retrieve the message. This paper proposes a system where image steganography is combined with encryption by hiding not only a message inside an image, but also the means to extract and decrypt the message. An executable program is hidden inside the image that functions as a decryptor, enabling the receiver to be oblivious to the encryption algorithm used.

1 Introduction

Encryption enables private communication by scrambling a message in such a way that it can only be recovered by using an appropriate decryption program in combination with an appropriate key. Encryption, however, suffers from a number of drawbacks – notably the fact that the mere presence of an encrypted message might be cause for suspicion.

Another drawback of encryption is the limitations that have been enforced by certain governments [1]. The use of encryption – and even the possession of an encryption algorithm – is illegal for ordinary citizens in some countries. This often implies that a traveler has to delete any encryption software when entering a country and is only allowed to acquire and install it again after leaving that country. Additional issues of encryption often imply that the receiver needs a number of decryptors and may have to occasionally get rid of them and reinstall them. People who wish to communicate in secret must thus find alternative ways of doing so.

Steganography, a technology used for hiding information in other information [2], is one such way. While steganography and encryption have their separate drawbacks, combining them result in a system that builds on the benefits of both. By first encrypting information and then embedding it inside an image, steganography adds another layer of security to encryption. An eavesdropper will first need to identify the embedded information, then extract the information and then finally decrypt it to use the secret.

Unfortunately, there is a drawback to this combination, namely the amount of overhead. With single encryption, as with single steganography, the receiver only has to have knowledge of the encryption, or steganographic, algorithm used to obtain the message. However when combining encryption and steganography, the receiver needs to not only have knowledge of how to decrypt the information, but also of how to extract it. This brings us to a problem similar to the cryptographic software distribution problem, where the software needed to decrypt the message has to be communicated to the receiver along with the encrypted message, making it harder to ensure the confidentiality of both.

This paper presents a solution to this problem by not only embedding the encrypted message in the image, but to embed the software to decrypt the message along with it, using steganography to distribute the decryptor on demand.

The remainder of the paper is structured as follows: Section 2 provides the reader with a brief overview of image steganography since it is a lesser known technology than encryption. Section 3 looks at the proposed design of the system. In Section 4 the advantages and the potential weaknesses of the proposed system are discussed and in Section 5 a conclusion is reached.

2 Overview of Steganography

Although many different cover mediums can be used for embedding information, images are the most popular mainly because of the redundancy created in the way that digital images are stored. In an environment where the Internet is used, images are also a common multimedia format, making it an ideal carrier for information, while reducing suspicion.

Image steganography techniques can be divided into two groups: those in the Image domain and those in the Transform domain [3]. Image – also known as spatial – domain techniques embed information in the intensity of the pixels directly and encompass bit-wise methods that apply bit insertion and noise manipulation.

For the Transform – also known as frequency – domain, images are first transformed, then the message is embedded in the image and they involve the manipulation of algorithms and image transforms [4]. These methods hide information in more significant areas of the cover image, making it more robust [5].

The simplest and most common technique for embedding information in images is called the Least Significant Bit (LSB) technique [6]. The least significant bit (the 8^{th} bit) of some or all of the bytes inside an image is replaced with a bit from the secret message. When using a 24-bit image, a bit of each of the red, green and blue colour components can be used, since they are each represented as a byte. In other words, one can store 3 bits in each pixel. For example, 3 pixels from a 24-bit image can be as follows:

```
(00101101    00011100    11011100)
(10100110    11000100    00001100)
(11010010    10101101    01100011)
```

When the number 200, which binary representation is `11001000`, is embedded into the least significant bits of this part of the image, the resulting pixels are as follows:

```
(00101101    00011101    11011100)
(10100110    11000101    00001100)
(11010010    10101100    01100011)
```

Although the number was embedded into the first 8 bytes of the grid, only the 3 underlined bits needed to be changed according to the embedded message. On average, only half of the bits in an image will need to be modified to hide a secret message using the maximum cover size [7]. Since there are 256 possible intensities of each primary colour, changing the LSB of a pixel results in small changes in the intensity of the colours. These changes cannot be perceived by the human eye - thus the message is successfully hidden.

3 System Design

The basic idea behind the proposed approach is to use steganography as a means of communicating secret, encrypted information along with the decryptor program.

3.1 System Specification

The system is divided into two phases: the embedding phase and the extracting phase.

Embedding Phase. The embedding phase is responsible for encrypting the secret message and embedding it into the image. Although any steganographic algorithm can be used, for the purposes of this research LSB will be used as an example together with the bitmap (.BMP) image file format. Knowledge of the encryption algorithm used is not imperative at this stage of the discussion.

The system consists of four algorithms. These algorithms are shown in Fig. 1. The first algorithm, the message-encryption algorithm, is simply used for encrypting the message and depends entirely on the encryption algorithm used.

The message-extraction algorithm is the algorithm used to extract and decrypt the message at the receiver's end. This algorithm is not explicitly used in the embedding phase, but has to be embedded into the image along with the message. This algorithm can either be in source code or converted to an executable (.EXE) program. There are advantages and disadvantages to both approaches that will be discussed later on in the paper.

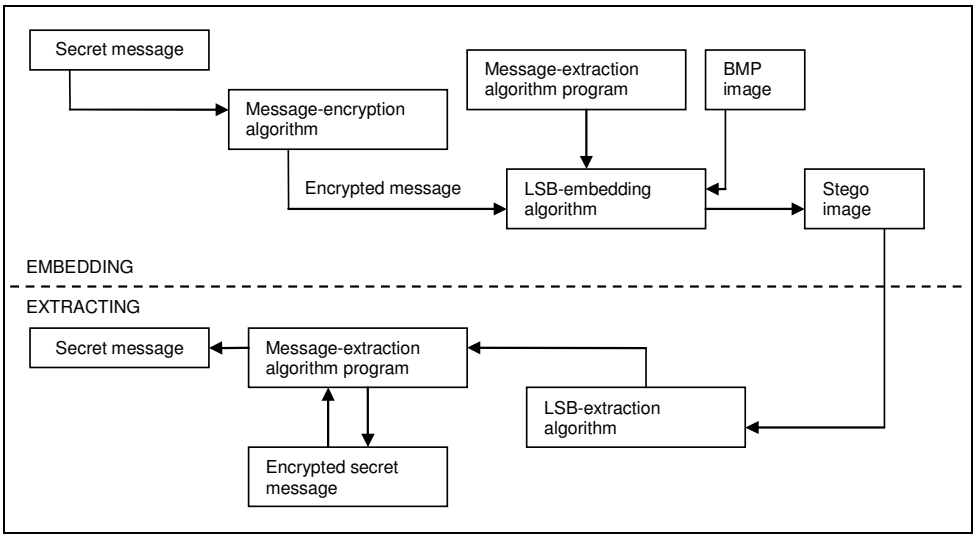

Fig. 1. System Design

The third algorithm, the LSB-embedding algorithm is used to hide the encrypted message along with the message-extraction algorithm. An inverted version of the LSB-embedding algorithm, the LSB-extraction algorithm, has to be communicated to the receiver through secure channels.

A specific format is specified for embedding information in the LSB-embedding algorithm.

`<filename.extension>$<4 bytes program size embedded in 32 bytes><message-extraction algorithm program>`

- The first 52 bytes of a BMP image consists of header data and cannot be modified
- The receiver will need to execute this program and since he will have no knowledge of the type of file beforehand, his ability to do so will directly depend on the inclusion of the filename and format. For this reason, after the first 52 bytes of the image, the filename of the embedded program, with special regard to the file extension, is hidden. Using LSB the embedding of the filename will start in the 53rd byte and continue in the following sequential bytes.
- The filename is followed by a dollar sign to indicate the end of the filename
- Since the receiver is unaware of the type of information to expect, he will also have no knowledge of the amount of information to extract. For this reason 4 bytes are used for storing the size of the message-extraction algorithm. If an attempt is made to extract more information than is actually embedded, the embedded program will not be able to execute accurately.

- Finally the message-extraction algorithm is embedded

LSB-embedding-algorithm. Let I be the carrier image with I' the image converted into binary. Each pixel in I is denoted as I'_i with i the pixel number. Each pixel consists of three colour components denoted as $I'_{i,RED}$, $I'_{i,GREEN}$ and $I'_{i,BLUE}$.
Let S be the secret message, converted into binary, S', and encrypted using the message-encryption algorithm resulting in $E(S')$.
Let P be the message-extraction algorithm program, converted into binary, P' with each bit denoted as P'_x where x is the bit number.
Let N be the filename of the message-extraction algorithm program, converted into binary N' with each bit denoted as N'_y, where y is the bit number

Calculate the size of the program P' in bits, denoted as F and converted into binary, F'
Set the value of i to 53 and **`For`** each bit in N'
Replace the LSB of the next pixel's I'_i three bytes as follows
    Replace the LSB of $I'_{i,RED}$ with a bit from N'
    Replace the LSB of $I'_{i,GREEN}$ with a following bit from N'
    Replace the LSB of $I'_{i,BLUE}$ with a following bit from N'
    Increment i
Convert the **$** sign into binary, D'
`For` the next 8 bytes of I'
    Replace the LSB of the next byte in I' with a bit from D'
`For` the next 32 bytes of I'
    Replace the LSB of the next byte in I' with a bit from F'
`While` not the *end-of-file* of P'
    Replace the LSB of the next byte in I' with a bit from P'
`While` not the *end-of-file* of $E(S')$
    .
    . *(depends on the manner in which E(S') is hidden)*
    .

After the message-extraction algorithm is embedded, the message can be embedded in a number of different ways. The least secure way would be to continue from the last byte of the message-extraction algorithm, since it would make the message easier to locate in the event of discovery of the decryptor program. If someone were to uncover the program, it would not necessary mean that they would suspect that there is more information embedded in the image. It would thus be wiser to use a different method to embed the secret message. One can either start embedding from the end of the image, or use selective LSB and only use a predetermined sequence of bytes. There are more possibilities that can be explored.

Extracting Phase. At the receiver's end, the extracting phase is where the program is extracted and executed to extract and decrypt the message. Two of the four algorithms are used in the extracting phase.

Using the LSB-extraction algorithm obtained from the sender, the message-extraction algorithm is retrieved from the image and stored in the appropriate file. Depending on whether the program is in source code or an executable program, the program can either be compiled and executed or simply executed. The program will receive the communicated image as input, locate and extract the message bits, and decrypt it.

LSB-extraction algorithm. Using the same definitions as the LSB-embedding algorithm, the following:

Set the value of i to 53
While N_y is not the **$** character
    Read in the LSBs of 8 bytes of I' at a time
    Convert the bits into ASCII and store in N_y
For the next 32 bytes
    Read in the LSBs of a byte of I', store it in F' and convert it into an integer number F
While F >= 0 do
    Read in the LSBs of 8 bytes of I' at a time
    Convert the bits into ASCII and store in P_x
    Save P in a file called N

3.2 System Considerations

The efficiency and functionality of the system can be measured with regards to invisibility and payload capacity. Invisibility being the first and foremost requirement since the strength of a steganographic system lies in its ability to be unnoticed by the human eye [8]. The invisibility of the system will depend entirely on the steganographic algorithm used, since some algorithms are more successful at unperceivable data hiding than others.

Payload capacity is the amount of data that can be stored in an image [8]. This is especially important in deciding which encryption algorithm to implement, since a very complex algorithm would result in a large program file that would not be able to fit into an image of reasonable size.

3.3 Prototype construction

Several prototypes were developed to implement the proposed system. Usually simple encryption algorithms were used, since the prototypes were developed to test the feasibility of implementing the proposed system and not the strength of the encryption. A comparison of two example prototype implementation is given in Table 1.

Table 1. Comparison of prototype implementations

	Embedded program	Embedded program function	Message size	Encryption	Embedded program size	Minimum image size	Payload capacity
Project A	Java class	Used for message encryption and decryption	150 bytes	Built-in DES function	4.6 KB	118 KB (200 x 200 pixels)	4%
Project B	Java class	Used for message extraction and decryption	150 bytes	Permutation and XOR	2.8 KB	30 KB (100 x 100 pixels)	9%

3.4 Source code or executable program?

Whether the embedded code has to be source code or an executable program, will depend on a number of factors. The advantages and disadvantages for use in the proposed system will need to be investigated.

When using source code, the receiver will be able to examine the embedded program. This could be useful when the sender wishes to communicate not only the message but also a specific encryption algorithm that he might want to use for future communications. This would mean that the sender need not send the decryptor program to a specific party in every following communication.

Another advantage for the receiver being able to examine the source code before executing it, is that there is always the possibility that the program in the image originates from a malevolent sender that might embed a malicious program in the image. When the program is an executable the receiver has no option but to blindly execute the program, having no idea what the program will do to his computer. The solution to this is a trust issue and will amount to the receiver trusting that the sender has not included any code that might damage his computer system. Alternatively a similar approach to a Java sandbox can be used to ensure that executable code does not gain access to the receiver's computer resources [9].

Depending on the programming language and compiler used, it is a very complicated task to decompile an executable program, in other words to retrieve the original source code from an executable program [10]. This can be made even more difficult when using code obfuscation, which is a technique for deliberately making source code difficult to read [11]. Should the nature of the sender/receiver relationship call for the confidentiality of the encryption algorithm itself, an executable program would be more suitable. All the

receiver can do is to execute the program and receive the decrypted message, without being able to gain knowledge of how the message was encrypted or decrypted.

The risk of discovery also plays a role in deciding whether to embed source code or an executable program in the image. Due to its nature, executable code gives the impression of being more like random data than source code, making it more difficult to notice should someone be looking for hidden information. Source code, being a close resemblance to natural language, is more prone to statistical attacks.

Finally an advantage of using an executable program over using source code is concerned with platform independence. An executable program will be able to execute on any platform, while some platforms might not be able to compile and execute certain source code. Along the same lines, communicating source code to a receiver is based on the assumption that the receiver actually possesses the correct compiler software to compile and execute the source code. This might not always be the case.

4 Advantages and Potential Weaknesses of the Proposed System

The concept of combining encryption with steganography in such a way to hide not only an encrypted message in an image, but also the decryptor program, holds many advantages over other forms of secret information communication. Unfortunately there are also potential weaknesses to the system.

4.1 Advantages of the Proposed System

The main advantage that the proposed system offers is by combining encryption and steganography you also combine their individual benefits. Cryptography mainly provides confidentiality to secret information, but can also provide authentication and nonrepudiation in some cases [12]. Steganography can also provide confidentiality of information, as well as privacy [13]. Steganography also provides security through obscurity, not always a good thing, but can be seen as a positive aspect in this case, since it is not the only means of security [14]. Importantly the proposed system provides a way of combining the two disciplines without increasing the amount of overhead used from the amount of overhead that a single encryption, or steganography, transaction would require.

A rather debatable advantage is that the proposed system makes provision for the use of proprietary encryption algorithms. Proprietary encryption algorithms are in most cases considered to be weak [15], since many get compromised due to inefficient algorithms. This aspect set aside, there are still many companies and individuals that prefer to use their own proprietary encryption algorithms to standard encryption algorithms. In the proposed system the fact that the algorithm is hidden inside the image increases the security of the algorithm and makes the distribution of the decryptor software more secure.

Another advantage of the proposed system is that it applies the diversity of defense concept [14], since it makes use of various layers of security. A lower security level steganographic algorithm can be used to embed the program and a higher security level steganographic algorithms can be used to embed the message.

4.2 Potential weaknesses of the proposed system

The first and most obvious risk to the proposed system is the fact that the decryptor and the encrypted message are stored in close proximity to one another. There are two possible solutions to this potential problem: Firstly one can divide the decryptor and the message between two different images. Embed the decryptor program in one image and embed the encrypted message in another and communicate them separately. As long as the decryptor program is capable of extracting and decrypting the message from the separate image file, the system will still function correctly. The second solution is to make use of cryptographic keys in the encryption of the message. Should someone try to execute the decryptor he will still need the secret key. Both of these solutions however will create more overhead, since more information needs to be communicated beforehand.

Another potential weakness lies in the way that the filename and file size are stored. Should an executable program be used for reasons of randomness, the filename and file size will still need to be in plaintext. This could provide valuable insight to an attacker who is trying to figure out what the true purpose of the hidden information is. A possible solution to the problem is to first encrypt at least the filename with a different encryption algorithm before it is embedded into the image. This approach however will create more overhead since the receiver must now again have knowledge of the encryption algorithm used in order to decrypt the filename.

Ultimately there exists a trade-off between the amount of overhead involved and the amount of security. More security could mean more unnecessary overhead, while less overhead will result in less security. It will ultimately depend on the desired level of security.

5 Conclusion

In trying to overcome the limitations that both encryption and steganography entail, a system is proposed that combines the two technologies in such a way as to minimise the amount of overhead used. This is done by embedding the decryptor program in the image along with the encrypted message.

The advantages that this approach offer include confidentiality and privacy of not only the secret message, but also potentially of the encryption algorithm. This results in other benefits that can be obtained, for example the secure use of proprietary encryption algorithms.

There are potential weaknesses to the system – most of their solutions include more overhead – and this brings about a trade-off between overhead and security. Ultimately, in whatever way the problems are dealt with, the proposed system will still involve less overhead than any similar security level combination of encryption and steganography.

References

[1] Dunbar, B., "Steganographic techniques and their use in an Open-Systems environment", *SANS Institute*, January 2002

[2] Jamil, T., "Steganography: The art of hiding information is plain sight", *IEEE Potentials*, 18:01, 1999

[3] Silman, J., "Steganography and Steganalysis: An Overview", *SANS Institute*, 2001

[4] Johnson, N.F. & Jajodia, S., "Steganalysis of Images Created Using Current Steganography Software", *Proceedings of the 2nd Information Hiding Workshop*, April 1998

[5] Wang, H & Wang, S, "Cyber warfare: Steganography vs. Steganalysis", *Communications of the ACM*, 47:10, October 2004

[6] Johnson, N.F. & Jajodia, S., "Steganalysis: The Investigation of Hidden Information", *Proceedings of the IEEE Information Technology Conference*, 1998

[7] Krenn, R., "Steganography and Steganalysis", http://www.krenn.nl/univ/cry/steg/article.pdf

[8] Morkel, T., Eloff, J.H.P. & Olivier, M.S., "An overview of Image Steganography", *Proceedings of the Information Security South Africa (ISSA) Conference*, 2005

[9] Rubin, A.D. & Geer, D.E., "Mobile Code Security", *IEEE Internet Journal*, December 1998

[10] Linn, C. & Debray, S., "Obfuscation of Executable Code to Improve Resistance to Static Disassembly", *Proceedings of the 10th ACM Conference on Computer and Communications Security*, 2003

[11] "Obfuscated code", *Wikipedia online encyclopedia,* http://www.wikipedia.org/wiki/Obfuscated_code, accessed on 6 July 2007

[12] Tudor, J.K., "Information Security Architecture: An Integrated Approach to Security in the Organization", *Auerbach Publications*, 2001, *book*

[13] Artz, D., "Digital Steganography: Hiding Data within Data", *IEEE Internet Computing Journal*, June 2001

[14] Conklin, A., White, G.B., Cothren, C., Williams, D. & Davis, R.L., "Principles of Computer Security: Security+ and Beyond", *McGraw-Hill Technology Education*, 2004, *book*

[15] Schneier, B., "Security in the Real World: How to Evaluate Security Technology", *Computer Security Journal*, Number 4, 1999