Design of a Data Hiding Application Using Steganography (April 2007)

Armin Bahramshahry, Hesam Ghasemi, Anish Mitra, and Vinayak Morada

Abstract— **This paper discusses the design of a data hiding application using steganography. Steganography is the term used to describe the hiding of data in images to avoid detection by attackers. Steganalysis is the method used by attackers to determine if images have hidden data and to recover that data. The application discussed in this paper ranks images in a users library based on their suitability as cover objects for some data. By matching data to an image, there is less chance of an attacker being able to use steganalysis to recover the data. Before hiding the data in an image the application first encrypts it. The application was built to adhere to the secure system development principles of defense in depth, open design, and psychological acceptability. The authors believe that the steganography method proposed in this paper and illustrated by the application is superior to that used by current steganography tools.**

Index Terms—**Steganography Application, Data Hiding, Image Ranking, Encryption Alternative**

I. Introduction

THIS document discusses our 412 design project, which was to design a data hiding application using steganography. The purpose of this project was to create a user friendly steganography application that allows users to hide private data in image files. Our goal was to make our steganography application less vulnerable to steganalysis than the existing steganography tools on the market.

Steganography is the process of hiding a secret message within another message. Steganography can be an invaluable tool in maintaining confidentiality, which is one of the three policies computer security is concerned with, along with integrity and availability.

The importance of steganography lies in the fact that it hides the very existence of the secret it is protecting. Attackers are threats because they attempt to damage or gain access to assets by taking advantage of these asset's vulnerabilities. Steganography makes the job of the attacker more difficult because the very existence of the asset is hidden

The importance of steganography in maintaining confidentiality can be illustrated with a simple example. Imagine two coworkers, Alice and Bob, are communicating with each other over the internet. Eve, an attacker, has access to this communication link, so she eavesdrops on Alice and Bob's communications. If Alice is asking Bob if he is free for lunch, then Alice probably does not mind if Eve reads this message. Thus, Alice can send her query to Bob along the communication link in plain text. However, if Alice is sending Bob confidential information, such as specifications for their company's latest project, then she probably does not want Eve to be able to read these messages. Therefore, Alice will likely encrypt her messages. A problem arises because the encrypted text is likely garbled, nonsensical data. Thus, Eve, even though she cannot read the encrypted messages, will know that Alice has a secret that she is sending to Bob. Eve can then take the encrypted message and attempt to crack it. This is a very real problem because as computational power increases, encryption is becoming easier to break [1]. However, if Alice uses steganography, and hides her secret message in a generic image file, then she can transmit her secret message to Bob without evoking Eve's suspicion. For instance, Alice can hide her secret message in a picture of her garden. She can then send the image, with the secret message hidden inside it, to Bob. Eve will think Alice is just sending Bob a harmless picture, so she will ignore that communication between Alice and Bob. Thus, Alice and Bob defeat Eve.

As mentioned in the example, attackers have more computing power now than ever before. This means that attackers are better able to break encryption algorithms and these capabilities will only increase in the future. DES, an encryption standard that was used by many national governments, successfully withstood attacks for many years since the mid 1970s. However, [2] mentions a cryptanalytic attack that can break DES in only a few minutes. Another example of a broken encryption algorithm is WEP. WEP was designed to provide confidentiality to users on wireless networks. [3] illustrates how WEP can be broken within hours. DES and WEP are examples of two encryption algorithms that were thought to be secure at the time of their design, but were broken in the future when attackers had more powerful computational resources. These examples prove that encryption is not enough to stop attackers from gaining access to confidential information. Steganography must also be employed to protect confidential assets from being compromised by attackers.

Steganography applications that hide data in images generally use a variation of least significant bit (LSB) embedding [4]. In LSB embedding, the data is hidden in the least significant bit of each byte in the image. The size of each pixel depends on the format of the image and normally ranges from 1 byte to 3 bytes. Each unique numerical pixel value corresponds to a colour; thus, an 8-bit pixel is capable of

displaying 256 different colours [5]. Given two identical images, if the least significant bits of the pixels in one image are changed, then the two images still look identical to the human eye [4]. This is because the human eye is not sensitive enough to notice the difference in colour between pixels that are different by 1 unit [4]. Thus, steganography applications use LSB embedding because attackers do not notice anything odd or suspicious about an image if its pixel's least significant bits are modified [4].

Unfortunately for every computer security strategy, there are attackers who develop countermeasures to defeat that security strategy. Steganography is no different; attackers combat steganography using steganalysis. Steganalysis is a process where attackers analyze an image to determine whether it has hidden data in it. A common steganalysis approach is to graph the pixel values of an image that is suspected of containing hidden data. Statistical analysis is then performed on the graphed pixel values [6]. The attackers hope to find anomalies in the statistical analysis of these images. These anomalies may indicate that the image contains a hidden message, and the anomalies may offer some insight into how to extract the hidden message. [6] claims that optimized steganalysis techniques can detect data hidden in an image, using regular LSB steganography techniques, with a probability ranging from 75% - 90%, depending on the size of the hidden message. Thus, using a good steganography algorithm is vital in hiding secret messages within images.

This report documents the design and development of our data hiding application using steganography. The goal of our application is to help users maintain their data's confidentiality. To achieve this goal, our application uses defense in depth. Not only does it hide the user's data within an image, but it also encrypts the user's data using the public key RSA algorithm. A user friendly GUI was incorporated to ensure psychological acceptability. The application does not rely on keeping its steganography algorithm a secret, nor is RSA a secret algorithm; thus, our application follows the secure programming principle of open design. To combat steganalysis, our application performs an analysis on the user's library of images. This analysis allows users to hide their data in the image that is least likely to be vulnerable to steganalysis.

Section II will describe related work that is currently available, followed by a description of how our algorithm works in Section III. Section IV discusses the results, limitations and future improvements of our application. Finally, we conclude the report in Section V.

II. Related Work

There are many steganography tools which are capable of hiding data within an image. These tools can be classified into five categories based on their algorithms: (1) spatial domain based tools; (2) transform domain based tools; (3) document based tools; (4) file structure based tools; and (5) other categories such as video compress encoding and spread spectrum technique based tools [7].

The spatial domain based steganography tools use either the LSB or Bit Plane Complexity Segmentation (BPCS) algorithm. The LSB algorithm uses either a sequential or scattered embedding schemes for hiding the message bits in the image. In the sequential embedding scheme, the LSBs of the image are replaced by the message bit sequentially (i.e. one by one in order, as mentioned in the introduction). In the scattered embedding scheme, the message bits are randomly scattered throughout the whole image using a random sequence to control the embedding sequence.

Two basic types of LSB modifications can be used for the embedding schemes described above. They are LSB replacement and LSB matching. In LSB replacement, the LSB of the carrier is replaced by the message bit directly. On the other hand, in LSB matching if the LSB of the cover pixel is the same as the message bit, then it remains unchanged; otherwise, it is randomly incremented or decremented by one. This technique, however, requires both the sender and the receiver to have the same original image, which makes LSB matching very inconvenient [7].

The current Steganography tools based on the LSB algorithms include S-Tools, Hide and Seek, Hide4PGP and Secure Engine Professional. These tools support BMP, GIF, PNG images and WAV audio files as the carriers [7]. Each of these tools has unique features. S-Tools reduces the number of colors in the image to only 32 colors. Hide and Seek makes all the palette entries divisible by four. In addition, it forces the images sizes to be 320x200, 320x400, 320x480, 640x400 or 1024x768 pixels. Hide4PGP embeds the message in every LSB of an 8-bit BMP images, and in every fourth LSB of a 24-bit BMP image. These applications are flawed because they do not analyze the image file after it has been embedded with data to see how vulnerable it is to steganalysis.

The transform domain based steganography tools embed the message in the transform coefficients of the image. The main transform domain algorithm is JSteg[7]. These applications can only work with JPGs because most other image formats do not perform transforms on their data.

The document based steganography tools embed the secret message in document files by adding tabs or spaces to .txt or .doc files [7]. These applications are limited because they only work with document files. They also cannot hide much data because there are a very limited number of tabs or spaces they can reasonably be added to a document. In addition, they are vulnerable to steganalysis because it is easy for an attacker to notice a document file that has been embedded with additional tabs or spaces.

The file structure based steganography tools embed the secret message in the redundant bits of a cover file such as the reserved bits in the file header or the marker segments in the file format [7]. These applications cannot hide very large data files because there are a very limited number of header or marker segments available for embedding hidden data.

There are also steganography tools based on video compression and spread spectrum techniques. The large size

of video files provides more usable space for hiding of the message. The spread spectrum technique spreads the energy of embedded message to a wide frequency band, making the hidden message difficult to detect [7]. These steganography tools are inconvenient because they require the users to send an entire video file every time they want to send hidden data.

III. Our Solution

As mentioned in the introduction of this paper, it is essential that a data bearing image be statistically and visually identical to the original image in order to avoid detection by an attacker. This was the goal we kept in mind while designing our data hiding application.

A. Data Hiding Algorithm

The key difference between our application and the other programs that implement LSB embedding is that our application ranks images based on their suitability as cover images for some data. This allows a user to pick an image suited for hiding particular data, which reduces the threat of steganalysis attacks. No other application we are aware of currently offers this functionality of matching an image to the data to be hidden.

In the application the user first specifies the data that they would like to hide, which can be in any file format. The application then encrypts this data using the recipient's RSA public key. Once the encrypted data is obtained, the procedure described in the following paragraph is repeated for each image in a user's image library.

Each bit of the encrypted data is compared to the least significant bit of the pixel bytes in an image. The comparisons are made starting from the first byte in the image until the last byte that permits all the data to be hidden in that image. The application cycles through the pixels of the image looking for the block of bytes that results in the least number of LSB changes. The image is then given a rank based on the percentage of least significant bits that match the encrypted data bits. Consider, for example 10 bits of encrypted data that need to be hidden in an image with a bit pattern of 10000000001. If some block of bytes in the image has least significant bits with a pattern 1000000011, this would result in the image receiving a ranking of 90%, because nine of the ten bits are an exact match.

Each image in the user's library is ranked as described above and the user is presented with this list of ranked images. The user is then free to choose which image to use to hide the data. The application does not automatically select the highest ranked image. The reason this final choice is left to the user is because while an image might be most suited to hiding the data, the image may not be one you would like to share.

Figure 1 shows a screen shot of the graphical user interface of the application. As can be seen the interface is intuitive and very simple to use. This design was selected to ensure that the application was usable by a lay person interested in a more secure way of communicating. This is keeping with the principle of psychological acceptability of secure software design.

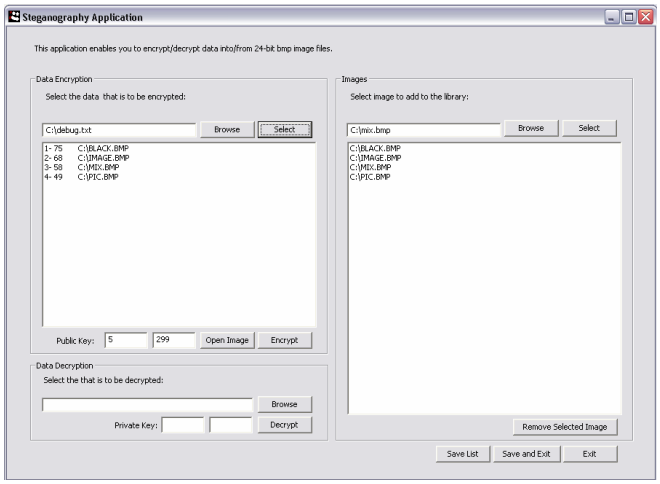

Figure 1: GUI Screen Shot

B. Encryption Algorithm

As mentioned previously, the data to be hidden is first encrypted using the RSA public key algorithm. Encrypting the data before hiding it provides defense in depth, and makes the job of the attacker more difficult if their goal is to recover the secret data.

The application uses the RSA algorithm for two reasons. First, by using a public key algorithm the need for a private shared key between the sender and recipient of the data is eliminated. Shared keys are impractical because they require a secure way of distributing the key to every person who you may want to communicate with. A public key for a person can be distributed fairly easily by publishing it on a website, or by emailing it to people you expect would need to send you secret information. Second, the RSA algorithm is also widely known and demonstrably secure if large enough prime numbers are used to generate the keys. Using an algorithm such as RSA which is public knowledge is in keeping with the principle of open design of secure software systems. Adhering to this principle was also the reason we chose not to use our own encryption algorithm.

IV. Discussion

A. Results

This section discusses the results of using our application to hide data in an image and corroborates the theory on which our algorithm is based. The cover image used to hide data is shown in Figure 2.


Figure 2: Original Image

The data hidden in the file is the string ‘Meet me at the park at noon’ encrypted using a public key of (5, 299). The image with the hidden data is shown in Figure 3. This particular cover image was chosen because it was ranked at 84% by our application which indicates that 84% of the least significant bits in the image matched the bits of data to be hidden.


Figure 3: Image with Hidden Data – High Rank

As can be seen these images are visually identical. The least significant bits of these images were also analyzed using a program called StegAlyzerSS developed by the Steganography Analysis and Research Center. This program costs $2495 to license and can be used to detect and analyze images that have hidden data. The LSB enhancement of the original image and that of the image with the hidden data are shown in Figure 4 and 5 respectively.

Figure 4: LSB of Original Image

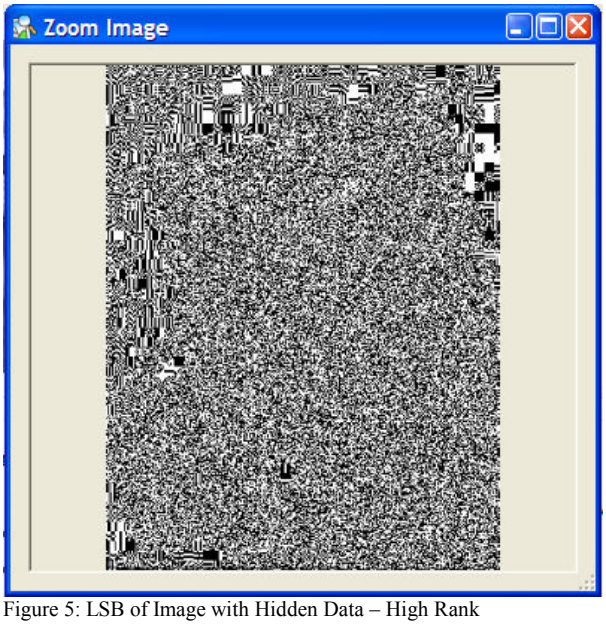

Figure 5: LSB of Image with Hidden Data – High Rank

As can be seen these LSB images are almost identical. StegAlyzerSS uses the LSBs of the image to analyze them for hidden data.

If the data to be hidden is changed to ‘This is not a good string to hide’ and the same public keys are used, the resulting image with the hidden data and its LSB enhancement are shown in Figure 6 and 7 respectively. With this string the image was ranked at 42% by our application.

Figure 6: Image with Hidden Data - Low Rank

Figure 7: LSB of Image with Hidden Data - Low Rank

As can be seen from the figures above, the image is almost identical to the original visually. However, comparison of Figures 4 and 7 indicated there is a perceptible difference in the LSBs of the two images. This makes the image in Figure 6 more susceptible to steganalysis. This corroborates our hypothesis that an image should be chosen as a cover based on its suitability to hide particular data. Thus, our application is an improvement over existing works because it gives users statistical information regarding how well their data can be hidden within a given image.

B. Limitations and Future Improvements

At the end of this project, we have a much wider view of the current state of steganography technology and the functionalities provided by current tools. This section discusses the limitations of our application and possible future improvements.

Our application currently uses the public key RSA algorithm to encrypt the data. The problem with using RSA is that there is currently no infrastructure in place for ensuring integrity of an individual's public key. Hence, we feel pretty good privacy (PGP) would be a better alternative for the public key encryption in a future version of the application. This is because with PGP there is already an infrastructure in place for distributing a person's public key. Also, with PGP there is a 'web of trust' infrastructure to verify that a public key does in fact belong to the correct person. This prevents spoofing of an individual's public key by attackers.

Currently, our application only supports hiding data in BMP images. This is a limitation because most images shared by people today are in the JPG format. The act of sending a BMP image in itself could cause an attacker to be suspicious of the image. Thus, an important future improvement for our application would be extending its functionality to support hiding data in JPG images.

Our application currently cannot hide data that is too large to fit in any of the images in the users' library. If large amounts of data need to be shared between people, the application could be extended to support breaking the data up into pieces and hiding each of these pieces in a different image. In this case each image would contain a special byte, so that the receiver of the images would be able to re-assemble the data at the other end. This idea uses a similar approach to the "modes of operation" used to concatenate blocks of cryptographic data.

Another approach to solving this problem of hiding large amounts of data would be to enable data hiding in video files. Video files are usually significantly larger than images and can hence be used for hiding more data. However, our application would still be able to hide data in images, thus avoiding the inconvenience of forcing users to send video files every time they want to send any hidden data.

V. Conclusion

This paper introduced the concept of steganography and steganalysis as well as the methods for carrying these out. It also presented the authors' application which was demonstrated to be more secure than current applications against statistical attacks commonly used in steganalysis. Recommended future improvements for the application were presented in the last section of the paper.

We believe that steganography when combined with encryption provides a secure means of secret communication between two parties. Our application, with its image analysis and ranking capability is a significant improvement on current steganography tools.

References

[1] J. Siegfried, C. Siedsma, B.J. Countryman, C.D. Hosmer, "Examining the Encryption Threat," *International Journal of Digital Evidence*, vol. 6, pp. 23-30, December 2004.

[2] E. Biham, A. Shamir. "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4, pp. 3-72, January 1991.
[3] A. Stubblefield, J. Ioannidis, A. D. Rubin, "A Key Recovery Attack on the 802.11b Wired Equivalent Privacy Protocol (WEP)," *ACM Transactions on Information and System Security*, vol. 7, pp. 319-332, May 2004.
[4] R. Chandramouli, N. Memon, "Analysis of LSB based image steganography techniques," *Image Processing*, vol. 3, pp. 1019-1022, October 2001.
[5] J. D. Foley, *Computer Graphics: Principles and Practicies*. Cornell: Addison-Wesley, 1996, pp. 3.
[6] J. Fridrich, M. Long, "Steganalysis of LSB encoding in color images," *Multimedia and Expo*, vol. 3, pp. 1279-1282, July 2000.
[7] Ming, Chen, Z. Ru, N. Xinxin, and Y. Yixian, "*Analysis of Current Steganography Tools: Classifications & Features"*, Information Security Centre, Beijing University of Posts & Telecommunication, Beijing, December 2006.