TJHSST Senior Research Project
Implementation of Steganographic Techniques
2006-2007

Danny Friedheim

June 11, 2007

Abstract

Steganography, the idea of hiding messages and data within other pieces of data, can be useful in many real-world applications alongside encryption and other code-writing methods. This project looks at the method of steganography known as least significant bit substitution, in which the pieces of a message are hidden within the actual binary data of a computer file. This particular application uses WAVE audio files as the carrier file, and allows messages of (practically) any length to be hidden within the sound data. The sound file itself looks identical and sounds the same to the human ear, so the existence of the message is very difficult to detect.

Introduction

This project implements a common steganographic technique known as least significant bit substitution. Steganography has many real-world applications in covert operations and communication. The development of an undetectable steganographic algorithm would be a huge breakthrough in the field and would have many implications. With this project, I propose to work towards that, starting with text messages in WAVE files.

Background

Steganography is the science of hiding data in a way that only the recipient knows of its existence. This differs from cryptography, where the existence of the data is known, but it is not readable. The process of steganography can be achieved with various algorithms designed to undetectably doctor an image, audio file, or other type of file. There is already a diverse field of research about steganography and its various applications in communicating secret messages. Commercial and open source programs that implement steganographic techniques include Stealth, stego, Wnstorm, Snow, FFencode, and many more. One example of an individual algorithm is the F5 technique for embedding messages in JPEG images. The algorithm changes the values of randomly generated bits by a very small amount, and is virtually undetectable by statistical analysis.

This project seeks to design a new program that works flawlessly to embed messages within WAVE audio files. The implications are that messages could be sent inside a file through a monitored path, and could even be intercepted. Without this same program on the receiving end however, a third party could not easily extract the message.

Development

This project uses an implementation of the least significant bit substitution algorithm to hide text messages within WAVE audio files. The result is a command-line program in C++ that acts as both the insertion and extraction mechanism. The program is run initially with a WAVE file and a text message as input, and it outputs a doctored WAVE file with the message hidden inside it. Then, the program is run with just the doctored file as the input, and the message is extracted and displayed. Therefore, different users who each have a copy of the program would be able to encode and decode each other's WAVE files. However, without the program, it becomes very difficult to even determine the presence of a hidden message, let alone extract it.

A few limitations were taken into account before implementing the algorithm. The most major restriction was time, because I knew I only had a few months to work with. This affected which goals I set and ultimately the final program, as some things had to be simplified to be finished in time. Another limitation was memory, but this ended up not coming into play. The algorithm is very fast and reasonably efficient, so memory was not a real limitation in actual implementation. However, if the program were to expand and start encoding extremely long messages within very large WAVE files, more efficient memory management might be necessary.

For this project, WAVE audio files were chosen as the carrier files because of their relative ubiquity on the Internet and their simple method of checking results (listening for differences in files before and after message insertion). Another possibility would have been JPEG images, but there has been extensive research done in this area already.

The actual method of inserting messages within the WAVE filesknown as least significant bit substitutionis a relatively simple concept. It runs on the

principle that if the least significant bit (or byte in this case) is overwritten with new data, the change will be undetectable to the human ear, yet will allow storage of data within the audio file. This program first stores an integer in the WAVE file header describing the length of the message being inserted. Then it goes through each sample of audio data in the file (4 bytes at a time) and overwrites the first byte with a character of the message text. The result is a working WAVE audio file only 4 bytes bigger in size (due to storage of the message length) with a text message cleverly hid within the data. In this project, the message can be either a short string written on the command-line at runtime or a text file containing a longer message.

The process of message extraction is essentially the opposite of insertion. First it looks at the length of the message (stored in the file header). With that information, it knows how far into the data to look for characters of the message. The first of each group of 4 bytes is recorded until the length has been reached, signaling that the complete message has been discovered.

Each of these methods is contained within a single C++ program. They can be accessed using command-line arguments such as (-i) for inserting a message and (-x) for extraction.

The success of the algorithm is confirmed in a few different ways. First of all, the existence of a message within the doctored file can simply be proved by running the extraction method. This displays the message found within the file, showing that it is indeed hidden. However, another aspect that must be confirmed is that the WAVE file still looks and sounds the same. By playing the files and comparing their sound (by ear) as well as visually inspecting the waveforms (in an audio editing program), we can ensure that the doctored files are indistinguishable, at this level, from the originals.

After completing the insertion methods for short, command-line messages I added a second option; inserting a message from a text file. This allowed me to test much longer messages and prove that any length of message could be added as long as there was enough data to cover it. Since each character of the message is inserted into 4 bytes of the sound data, a message up to the length of the data divided by 4 can be inserted without error.

```
dfriedhe@okonokwo ~/seniorresearch $ ./a.out sample2.wav -i2 MESSAGE!
sample2.wav MESSAGE!
RIFFWAVE 424100 WAVE
stereo
sample bit rate 44100
data chunk reached
datasize: 423488
data read successfully
datsize confirmed
datlen423456
message written to output3.wav
dfriedhe@okonokwo ~/seniorresearch $ ./a.out output3.wav -x2
Message found:
MESSAGE!
```

Figure 1: Sample program output during insertion and extraction

Discussion

The purpose of the project was to develop a program for insertion (and extraction) of text messages within WAVE audio files. The steganographic method used, known as least significant bit substitution, has been implemented with no undesirable side effects on the outputted WAVE file. The outputted file sounds the same as the original and is only 4 bytes bigger in file size. Therefore, it would be very difficult for an undesirable source to discover that the file contains a hidden text message. The relatively informal methods of confirming the success of the program have thus proved it to be working well. In addition to comparing file sizes before and after insertion of a message, the sound of the files has also been compared by ear. There is a slight hiss over the outputted file when a very long message (like the Declaration of Independence) is embedded in the WAVE file, but for shorter applications it sounds identical to the original file.

Another way the WAVE files were compared before and after message insertion was by looking at the actual waveforms of the files in an audio editing program. The screenshots below show that the original file (breakbeat.wav) is practically identical to the outputted file containing a message (output3.wav) even on a very small scale. Though I only did visual inspection to confirm that the file is not being significantly changed by the message, I found that sufficient to show success of the insertion algorithm.

Throughout the project I had hoped to be able to find a statistical test that I could run on the program as a form of steganalysis. Steganalysis, es-

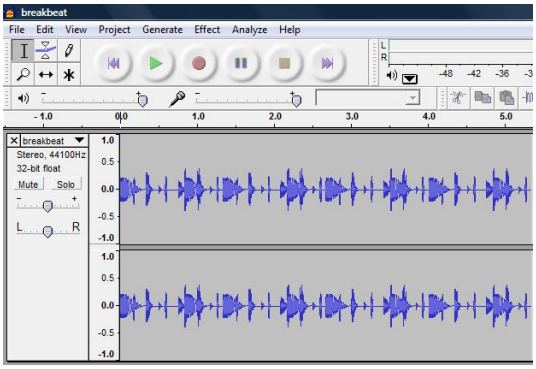

Figure 2: Waveform of "breakbeat.wav" before message insertion

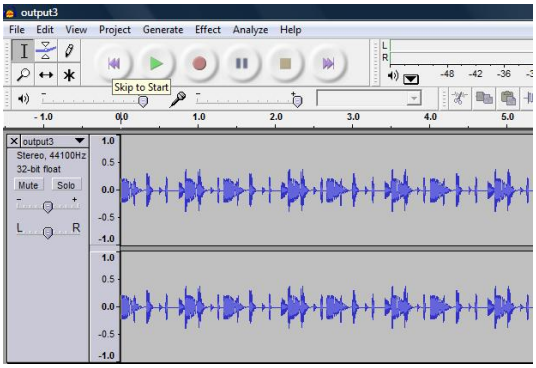

Figure 3: Waveform of "output3.wav" with Declaration of Independence inserted

sentially the opposite of steganography, is the act of attempting to discover hidden messages in files through statistical calculations and other methods. Most of the programs that are out there currently deal with JPEG images and a few other image file types, so I hoped to develop my own statistical analysis tool to discover files with messages hidden inside. Unfortunately I was not able to even get a start on this because of time constraints. Thus, working on it would be a good follow-up project that could deepen the understanding of steganography and steganalysis.

Conclusion

I have found that the least significant byte substitution method of steganography is a valid and reliable method of hiding text messages within WAVE audio files. However, it would be easy to also expand to include many other file types as carriers as well.

The success of the algorithm was proved with visual inspection of waveforms, comparison of sound files by ear, and by inspection of file sizes of both the original sound files and the outputted files containing messages. This was a successful showcase of the power of steganography and the simple applications that it can have.

On a larger scale, there are even more important applications of steganographic techniques such as LSB substitution. They are especially useful in espionage and military field operations because they allow covert communication to go undetected. These techniques could one day lead to Internet connections that are almost impossible to crack, without using the traditional methods of encryption that we all know about already.

Bibliography

- R.J. Anderson, F.A.P. Petitcolas, "On the limits of steganography", *Selected Areas in Communications, IEEE Journal on*, vol. 16, issue 4, pp.474-481, 1998.
- T. Aura, "Practical Invisibility in Digital Communication," Lecture Notes in Computer Science, Springer-Verlag, Berlin, vol. 1174, 1996, pp. 265-278.
- G.A. Francia, T.S. Gomez, Steganography obliterator: an attack on the least significant bits, *Proceedings of the 3rd annual conference on Information security curriculum development*, 2006. Pages 85-91.
- J. Fridrich, M. Goljan, D. Hogea, "Steganalysis of JPEG Images: Breaking the F5 Algorithm", *Lecture Notes In Computer Science*, vol. 2578, pp. 310-323, 2002.

- A. Habes, 4 least Significant Bits Information Hiding Implementation and Analysis, GVIP 05 Conference, 19-21 December 2005.
- N. F. Johnson, S. Jajodia, "Exploring Steganography: Seeing the Unseen", *IEEE Computer*, 1998.
- Noto, MP3Stego: Hiding Text in MP3 Files, SANS Institute, September 15, 2001.
- K. Rabah, Steganography-The Art of Hiding Data, *Information Technology Journal 3* (3): 245-269, 2004.