Applying Fuzzy Hashing to Steganography

Jeffery D. Dodson and Ambareen Siraj

***Abstract*—Steganography is a stealth technique used to hide messages inside of images, typically used for unauthorized covert communication. Detecting images that have hidden data inside of them has always been an interesting and challenging problem in cyber security. There are currently a handful of methods that exist for detection of such hidden messages in the form of data or images, which mostly require physical examination. In this paper, we describe a solution to automatic detection of Steganographic images with the novel application of the fuzzy hashing technique. In this paper, we discuss the methods used in this study, the experimental results, followed by a discussion on future research.**

***Index Terms*—Context triggered piecewise hashing, fuzzy hashing, image analysis, image processing, Steganography.**

I. Introduction

Image Steganography is a stealth technique used to hide message inside of image, typically used for unauthorized covert communication. While the original message itself will be altered at some level to hide the information of interest, efficient Steganography programs assures that the hidden information is undetectable by the naked eye. Steganalysis, which is the detection of such hidden information, is particularly challenging, as such solutions often require physical examination of the digital evidence or the use of very high compression techniques. In this research, we applied Fuzzy hashing for Steganalysis. Fuzzy hashing is a technique used to determine the extent of similarity between two entities.

This paper is structured in the following way. Section II discusses the background of the research and related work, including description of the fuzzy hashing application. Section III describes the experiments and their results. Section IV discusses future work regarding further improvements and application. Section V concludes the paper.

II. Background and Related Work

Steganography and fuzzy hashing are both very interesting and well-researched subject matters in cybersecurity. The following is a brief discussion of the two.

A. Steganography

Steganography is the act of hiding an image, text, or another file (stegotext) inside of another image, file, or text (covertext) [1]. Classically, it refers to using some kind of message as a cover to hide another secret message [2]. Steganography is primarily used for hiding text or image files inside of image files. The use of Steganography can be traced back to ancient Greece where text was hidden under writing tablets' wax [2]. The study of Steganography gained popularity in 1983 with Simmons' famous "Prisoner's Problem", where ciphertext had to be hidden with inconspicuous covertext [3].

For the purposes of this paper, Steganography will refer to hiding information (text or image) inside of an image. There are many different ways that Steganography can be implemented. One simple method of Steganography involves hiding information inside of the pixels of an image in the most inconspicuous pieces [4]. Often, encryption is used to generate a keystream based on a secret key to select appropriate pixels to hide information in [2]. More advanced methods also use algorithms that check pixels and their surrounding pixels to make sure that they can be used to appropriately hide information without being visibly noticeable [2]. When Steganographers anticipate compression prior to message transmission, such as for large jpeg images, to thwart prevention, they split up the hidden message to reside in multiple places or inside of the cover image's frequency domain [5], making covert communication feasible - even with compression.

There exist a vast number of options for different Steganography tools. Many of these tools are free or open source. The main differentiators between the tools are the different types of files that one can use to hide messages within and the actual messages to hide. There are tools that will only allow one to hide text inside of images, and vice versa. To investigate the effectiveness of the application of fuzzy hashing in Steganalysis, we wanted to use a versatile open source tool that would provide the capability to hide any file type inside of any other file type, and the tool we selected is: Stego Magic.

1) Stego magic

Stego Magic is a publicly available open source Steganography tool [6]. It can be downloaded as two executable files; one for hiding text files inside of other files and another for hiding any binary file inside of any other binary file. In essence, this allows one to hide any file type inside of any other file type. The two different executables behave in the exact same way. The program is easy to use. Both executables open up a window that is very similar to a command prompt window. As this tool works for all types and sizes of files [6], it makes it an ideal candidate for this experiment because we assume in our experiments that the size of the hidden file is unknown. In the real world, any size of file can potentially be hidden, and an effective Steganalysis approach should be able to work in all cases.

Manuscript received September 17, 2015; revised November 11, 2015.

The authors are with Tennessee Technological University, 110 University Drive, Cookeville, TN 38505 USA (e-mail: jddodson43@students.tntech.edu, asiraj@tntech.edu).

B. Fuzzy Hashing

Hashing is a cryptographic technique to compute a unique representation of a digital entity. Hashing allows one to uniquely identify an entity (except for rare collisions). Thus, to compare entities, either the computed hash matches with a known hash or it does not - there is no in-between or consideration of match to a certain degree. Fuzzy Hashing is a method of hashing that uses fuzzy logic to identify two entities as similar to some extent [7]. Fuzzy logic, based on the theory of possibility, deals with the notion of uncertainty by measuring the degree of truth [8]. This approach is effective when issues are uncertain, vague or incomplete.

Fuzzy hashing is more technically referred to as context triggered piecewise hashing (CTPH) [7]. Context triggered piecewise hashes are made up of a combination of two kinds of hashes, a piecewise hash and a rolling hash [9]. While a regular hashing algorithm creates a single hash for the entire file, piecewise hashing creates multiple hashes/checksums for the single file. In other words, piecewise hashing generates multiple hashes for certain sized segments of a file [9]. This can be thought of as block based hashing. Block based hashing is where one takes the input and divides it into equal sized blocks where the hashes of each individual blocks are calculated [7]. A rolling hash works by creating a pseudo-random value using the given input in a sequential fashion [9]. It is essentially a function that changes based on the last bytes it receives as input. Context-triggered piecewise hashing combines piece-wise and rolling hashing to create fuzzy hashes [9].

1) Ssdeep

Ssdeep is an application used in computer forensics for creating and comparing fuzzy hashes [9]. Ssdeep is based on the fuzzy hashing technique that was originally created for spamsum, an application to detect spam by identifying potential spam emails with similar features to known spam emails [7].

What makes ssdeep powerful is that it allows one to compare the context triggered piecewise hash values, called signatures, to each other to determine how similar they are. The algorithm determines a scaled weighted edit distance of the files after removing sequences from the block size of the signatures, which is the trigger value for the rolling hash [9]. This produces a match score as output, which can be thought of as a similarity percentage.

C. Related Work

As mentioned before, originally fuzzy hashing was applied for spam detection [9]. The spamsum algorithm was developed and successfully used by Dr. Andrew Tridgell to identify emails that are similar to known spam emails by checking for match values greater than 50 [9].

Breitinger and Baier offer some improvements for the fuzzy hashing methodology [7] and implement their own method for fuzzy hashing called bbHash, which is based on random Sequences and Hamming Distance" [10]. This method is more robust, allows for more precision, and allows for very small parts of files to be compared [10].

F. H. Al-Rubbaiy used fuzzy hashing for the concealment of images using Steganography [11]. Using a fuzzy approach to hiding text allowed for resilience against some common types of detection methods [11].

Fuzzy Hashing has been used successfully for the detection of malware [12]. While traditional hash methods failed to recognize files that are not exactly identical to files known to contain malware variants, fuzzy hashing was able to find files with similar hashes indicating variants of the malware in files [12].

III. Experimental Approach

We believe that fuzzy hashing can be an effective solution to the detection of steganographic images. Images that have information hidden inside of them should generate signatures/hashes, which may not be identical to the ones with no information hidden but can be similar to some extent. Moreover, this similarity would degrade to the extent of the size/nature of the hidden information.

In order to study this hypothesis, we used ssdeep and experimented with a number of different stegotext and covertext combinations. These included a wide range of sizes of images with different sizes of hidden text or images. Section A. explains the experimental setup, section B explains each experiment in detail, and then section C presents the results of the experiments with discussion.

A. Setup

The experiment consisted of a series of tests matching different fuzzy hashes against each other using ssdeep's file signature comparison function. First, different text files varying in sizes were created and then hidden inside image files with Stego Magic. The images also varied in sizes. Then the fuzzy hashes for the images were calculated using ssdeep and its recursive functionality, the –r flag, to insert all the hashes into one text file per experiment. Lastly, the actual file signature comparisons were executed between all of the hashes by using the –x flag for the ssdeep command. This produced match values, which can be thought of as similarity percentages.

1) Image selection

Four different popular images were selected for the experiments. The reason behind selecting these is that there is a vast amount of copies of these images available on the Internet. This means that if these images were used to hide information, they would likely not raise any kind of alarm due to familiarity. The images are tankman.jpg, sadkeanu.jpg, grumpycat.jpg, and water.jpg. Tank Man is a famous historical image, Sad Keanu and Grumpy Cat are both famous for being Internet memes, and Water is a rather popular high definition desktop background.

B. Experimental Setup

The first three experiments used images that had text files hidden inside of them via Steganography (using the Stego Magic tool). Six text files with increasing sizes were used. 1.txt was 6kb, 2.txt was 15kb, 3.txt was 23kb, 4.txt was 34kb, 5.txt was 56kb, and 6.txt was 130kb. The last two experiments used images that had other images hidden inside of them. Also, two images (space and water) were hidden a different number

of times to simulate different sized images. These experiments included seven test cases. One copy was hidden for the first test, two were hidden for the second test, three were hidden for the third test, and so on. A test using different images of various sizes could also be performed rather easily and would show almost identical results, as seen in our preliminary testing.

Fig. 1 depicts the basic layout of each experiment. The rounded rectangle represents the original image, double rectangles represent text files, and double rounded rectangles represent images with information hidden inside of them. Rectangles with two vertical lines represent programs used and hexagons represent match comparison results. Thus, at the top is the original image (covertext) with the six text files (stegotext). These files all feed into Stego Magic, which creates the six steganographed test images as output. The six test images and the original image are sent to ssdeep for match comparison and the results of these comparisons are shown at the bottom.

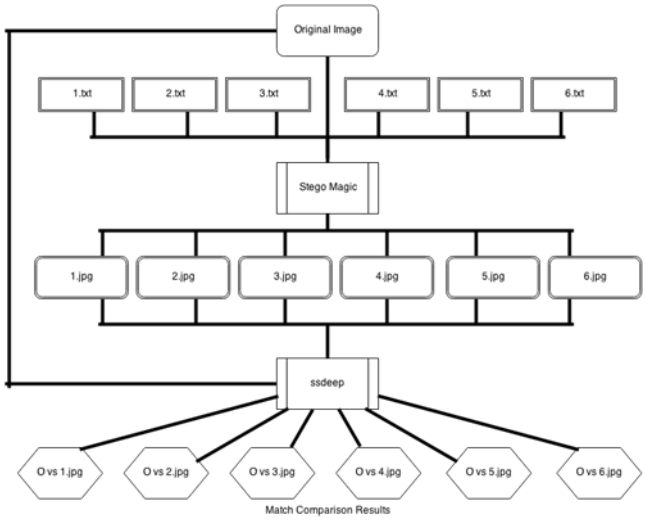


Fig. 1. Experimental setup.

Five experiments of the same setup were conducted for five original images of varying sizes.

1) Experiment 1

The first experiment compared the hashes computed by ssdeep for a medium sized image hiding a series of different sized text files inside of it. The image used for this experiment was tankman.jpg, a famous image of a Chinese protestor who stood in front of a series of tanks [13]. The original image measured 940x530 and was 65.4 kb in size. The test images were named 1, 2, 3, 4, 5, and 6 after their respective text tiles. The largest image, 6, ended up being 196 kb in size, a very plausible size for an image like this.

2) Experiment 2

The second experiment used a smaller image sadkeanu.jpg, a picture famous for being an internet meme [14]. The original image was 16 kb and was 480x360 pixels. The test images were named 1, 2, 3, 4, 5, and 6 as in experiment 1. 6.jpg ended up being 145 kb, which once again, was a plausible size and would not raise alarm on its own.

3) Experiment 3

The third experiment used a very large cover image with small text files hidden inside of it. The image was grumpycat.jpg, a picture also famous for being an internet meme [15].

4) Experiment 4

The fourth experiment consisted of an original image hiding other images inside of it via Steganography. It used the image water.jpg. This is a rather popular high definition background easily accessible via a Google search for "high definition wallpapers."

5) Experiment 5

The fifth experiment reused the image of Grumpy Cat in order to compare how the results would differ when images were hidden in an image versus when texts were hidden.

C. Results and Analysis

1) Experiment 1 (Various sized texts hidden inside medium sized image) results

Experiment 1 showed very promising results (Fig. 2). It showed that the all hashes compared against the original at varying levels of similarity where the similarity measures decreased with size of the hidden messages.

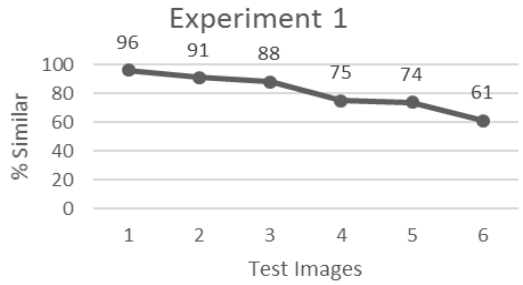


Fig. 2. Results of experiment 1.

2) Experiment 2 (Various sized texts hidden inside small sized image) results

Experiment 2 showed similar results to those of experiment 1 (Fig. 3). The hash comparison percentages all clearly showed that something was very different about these images when compared against the original. Since there was a major size difference between the cover image, which was very small, and the text file, 6.txt, which was the largest in the group, it caused the image to change so drastically that the result of the fuzzy hash comparison was below the threshold of similarity (hence the 0 in the chart). Another interesting point is that test images 4 and 5 both displayed similarity of 41% against the original, despite having different sized text files hidden inside. When compared to each other, the hashes of images 4 and 5 were 94% similar. This may be due to the transformation function of ssdeep dealing with precision.

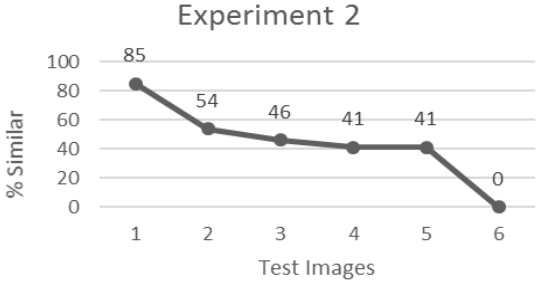


Fig. 3. Results of experiment 2.

3) Experiment 3 (Various sized texts hidden inside very large image) results

In Experiment 3 (Fig. 4), the hashes for all tests indicated they are 99% similar to the original. This may be due to lack of precision used in the ssdeep program. If one could examine files in finer precision, perhaps these changes would become apparent.

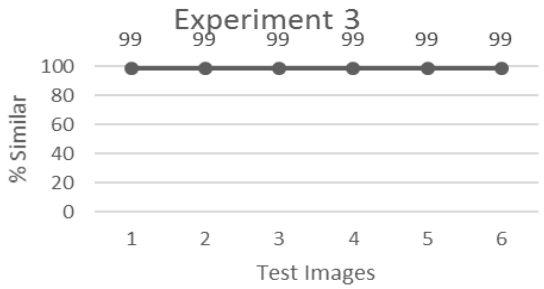


Fig. 4. Results of experiment 3.

4) Experiment 4 (Various sized images hidden inside medium sized image) results

As shown, there is significant dissimilarity between the fuzzy hashes (Fig. 5). Beyond test image 4, the similarity gets so insignificant that ssdeep does not even recognize them as similar.

Fig. 5. Results of experiment 4.

5) Experiment 5 (Various sized images hidden inside very large image) results

Experiment 5 showed very promising results (Fig. 6) in comparison with experiment 3 (Fig. 4). Both of these experiments used the same original image, and while experiment 3 showed that hiding small text files did not alter the very large original image very much, experiment 5 showed that hiding images resulted in very different hash values. The same overall trend of decreasing similarity for increasing hidden image size is seen here as well.

Interestingly, test images 4, 5, and 6 all compared at the same level of similarity, despite being of different sizes. This may also be related to how ssdeep handles precision.

Fig. 6. Results of experiment 5.

6) Overall results and analysis

As a whole, the experimental results were very promising in demonstrating that fuzzy hashing can be used as a viable method for detecting images with hidden messages (text or images). All experiments, with the exception of experiment 3, showed an overall trend of decreasing similarity when compared with increasing hidden file size. Fig. 7 combines the results of all experiments on one graph. Something to keep in mind that results could vary based on the Steganography program used. One improvement that could be made to these experiments is to enable the fuzzy hashing algorithm to compute with more precision such that slightest differences are taken into consideration.

One interesting observation about these experiments was the reporting of identical similarity measures, even if the hidden files were different. While a match value of 0 should warrant immediate attention (indicating that there is a very large file hidden inside of an image), other similarity measurements should also warrant further investigation.

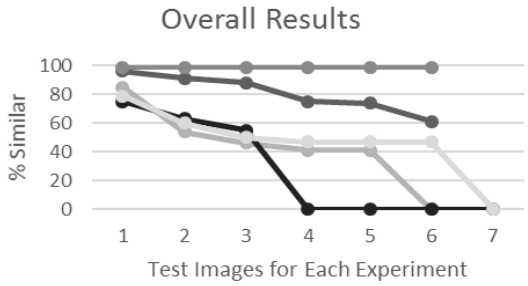


Fig. 7. Results of all experiments.

IV. Future Work

The approach described in this paper requires the original image for comparison. To align more with real world applications where images are likely to be intercepted and analyzed to detect covert communication, future research is ongoing to devise an automated technique that would allow one to scout similar images available in the Internet for comparison with image under investigation. In this regard, Reverse Image Searching tools, such as TinEye, have the potential to be used to find images that appear similar to the image in question. TinEye is able to retrieve images that have been edited in some way, such as being cropped or resized [16]. This implies that given a steganographic image, TinEye should be able to find candidates to compare with such that fuzzy hashing approach can be applied to detect and confirm suspicious steganographic images.

V. Conclusion

In this paper we have outlined a novel approach for the detection of steganogaphic images. The results of these experiments have shown that the fuzzy hashing method can be useful for Steganography detection and could become a great tool for computer forensics and cyber security.

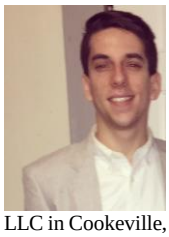
Acknowledgment

Our thanks to Jesse Kornblum for developing ssdeep and making it available to public for general use. We would also

like to thank Srikanth Ramesh for creating Stego Magic and making it available for free. Thanks to Dr. Elizabeth Dodson for her help in peer reviewing this paper.

REFERENCES

[1] A. Cheddad, J. Condell, K. Curran, and P. Mc. Kevitt, "Digital image steganography: Survey and analyses of current methods," *Signal Processing*, vol. 90, issue 3, pp. 727-752, March 2010.
[2] R. J. Anderson, "On the Limits of steganography," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 4, pp. 474-481, May 1998.
[3] G. J. Simmons. "The prisoners' problem and the subliminal channel," in *Proc. CRYPTO '83*, pp. 51-67, 1984.
[4] R. G. van Schyndel, A. Z. Tirkel, and C. F. Osborne. "A digital watermark," *IEEE Trans. Image Process.* vol. 2, pp. 86-90, 1994.
[5] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, "A secure, robust watermark for multimedia," *Information Hiding, Springer Lecture Notes in Computer Science*, vol. 1174, pp. 183-206, 1996.
[6] S. Ramesh. (Oct. 28, 2011). *Hide Data in Image, Audio and Video Files: Steganography.* [Online]. Available: http://www.gohacking.com/hide-data-in-image-audio-video-files-steganography/
[7] F. Breitinger, "Sicherheitsaspekte von fuzzy-hashing," M.S. thesis, Fachbereich Informatik, Hochschule Darmstadt, Darmstadt, Germany, 2011.
[8] J. Yen and R. Langari, *Fuzzy Logic: Intelligence, Control and Information*, Upper Saddle River, NJ: Prentice Hall, 1999.
[9] J. Kornblum, "Identifying almost identical files using context triggered piecewise hashing," *Digital Investigation*, pp. 91-97, 2006.
[10] F. Breitinger and H. Baier, "A fuzzy hashing approach based on random sequences and hamming distance," in *Proc. ADFSL Conference on Digital Forensics, Security and Law*, pp. 89-101, May 2012.
[11] F. H. Al-Rubbaiy, "Concealment of information and encryption by using fuzzy technique," Univ. of Al Mustansuraia, Baghdad, Iraq. 2011.
[12] D. French, "Fuzzy hashing techniques in applied malware analysis," *Software Engineering Institute Blog*, 2011.
[13] J. Makinen. (June 4, 2014). Tiananmen square mystery: Who was 'Tank Man'? [Online]. Available: http://www.latimes.com/world/asia/la-fg-china-tiananmen-square-tank-man-20140603-story.html
[14] Keanu Is Sad / Sad Keanu. [Online]. Available: http://knowyourmeme.com/memes/keanu-is-sad-sad-keanu
[15] Grumpy Cat. [Online]. Available: http://knowyourmeme.com/memes/grumpy-cat
[16] Idée Inc. (2014). *FAQ – TinEye.* [Online]. Available: https://www.tineye.com/faq



Jeffery D. Dodson is a master's candidate in the MBA program at Tennessee Tech University in Cookeville, Tennessee. He got his undergraduate degree in computer science with a focus in software and scientific applications at Tennessee Tech University in May 2015.

He currently works as a database programmer and informatics analyst for Cumberland Health Analytics, LLC in Cookeville, Tennessee. He is also currently a research assistant to Dr. Thad Perry at the Center for Healthcare Informatics at Tennessee Tech University in Cookeville, Tennessee. He is currently working on research involving data mining of healthcare claims to find new ways of grouping claims as well as text mining patient satisfaction surveys in order to create a more useful feedback form based on patient interests. He has previously done research in the field of cyber security on both the safety of QR Codes as well as steganography.

Ambareen Siraj is an associate professor at the Computer Science Department and serves as the director of cybersecurity education, research and outreach center at Tennessee Tech University. She teaches various security courses both at undergraduate and graduate level. Her research is in the areas of situation assessment in network security, secure communication in smart grid, security metrics and security education. She has authored/co- authored around thirty journal and conference articles in security. She also serves as the faculty advisor of Tennessee Tech Cybersecurity Club for students.