5-2022

Identifying Text File Similarities in Forensic Disk Images Using Fuzzy Logic

Mindy M. Wongsa
University of South Alabama, mmw1425@jagmail.southalabama.edu

IDENTIFYING TEXT FILE SIMILARITIES IN FORENSIC DISK IMAGES USING FUZZY LOGIC

A Thesis

Submitted to the Graduate Faculty of the
University of South Alabama
in partial fulfillment of the
requirements for the degree of

Master of Science

in

Computer Information Systems

by
Mindy M. Wongsa
B.S., University of South Alabama, 2019
May 2022

TABLE OF CONTENTS

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

LIST OF ABBREVIATIONS

CD ..... Compact Disc

CTPH ..... Context Triggered Piecewise Hash

DF ..... Digital Forensics

DVD ..... Digital Versatile Disc

MF ..... Membership Function

NB ..... Negative Big

NIST ..... National Institute of Standards and Technology

NM ..... Negative Medium

NS ..... Negative Small

PS ..... Positive Small

PM ..... Positive Medium

PB ..... Positive Big

ZE ..... Zero

ABSTRACT

Wongsa, Mindy, M., M. S., University of South Alabama, May 2022. Identifying File Similarities in Forensic Disk Images Using Fuzzy Logic. Chair of Committee: Michael Black, Ph.D.

Digital storage is evolving with the growth of technology. Individuals and corporations can access large amounts of digital storage, leaving digital forensics investigators with large amounts of data to collect and analyze in their forensic investigation cases. In addition, analyzing forensic disk images that contain hundreds of thousands of files can cause a problem with time since the investigators' workloads can vary based on how many cases they are assigned. Fuzzy logic provides a pattern recognition system that could assist in identifying patterns in data. The purpose of this study was to determine if fuzzy logic could reliably aid in identifying similarities between text files for digital forensics investigators analyzing large amounts of data in the preliminary stage of the investigation. In addition, this study was used to determine if fuzzy logic was unreliable in identifying similar text files at the 20% - 60% granularity levels. However, at the 70% - 90% granularity levels, fuzzy logic was reliable in identifying similar text files. Based on these two statements, fuzzy logic was not trustworthy overall since it could not correctly identify similar text files at all granularity levels.

CHAPTER I

INTRODUCTION

The growth of digital storage has made it possible for individuals and corporations to store large amounts of data on their storage devices and servers. Digital forensics is the applied science for identifying, collecting, organizing, and presenting evidence from a digital source in civil, criminal, or corporate cases (Zareen et al., 2013; Zawoad & Hasan, 2016). Hou et al. (2020) state that the National Institute of Standards and Technology (NIST) recommends dividing digital forensics investigations into four consecutive phases: collection, examination, analysis, and reporting. The analysis phase analyzes the acquired digital device by creating a forensic image copy of the original system using specified hardware and software tools (Abdullah et al., 2020; Varol & Sönmez, 2017).

The growth of digital storage can be an issue for digital forensics investigators during the analysis phase investigation. The information found on the devices during the analysis phase will need to be examined and analyzed to conclude a case based on the evidence present on the storage device (Caviglione et al., 2017). Digital crime is the use of any digital device to commit criminal activity. It can vary in different forms online and offline, including cyber violence, cyber trespass, data alteration, interception, and theft (Rao et al., 2020; Stratton et al., 2017). The evolution of digital crimes has made it challenging to detect criminal activity for an extended time. (Horsman, 2017).

1.1 Research Problem, Purpose, and Significance

This research aims to improve the analysis phase of the digital forensic investigation using a proposed algorithm for testing the reliability of fuzzy logic to identify text files with similarities. The analysis phase creates a basis for the trial and helps identify possible conclusions for the judicial authority (Varol & Sönmez, 2017; Yankson & Davis, 2019). A forensic disk image is a bit-by-bit copy of the original digital device that contains all the information and data needed for the analysis (Ali et al., 2005; Yang et al., 2020). Any evidence related to the crime is then analyzed further, including the possible incidents that could have occurred, the time of the incidents, solid and electronic evidence, and the type of software used (Varol & Sönmez, 2017; Yankson & Davis, 2019).

The growth of digital storage has made it challenging for digital forensics investigators to have the appropriate tools to analyze data (Caviglione et al., 2017). One of the challenges of digital forensics investigations is the increase of storage in digital devices and the speed needed to process these devices (Caviglione et al., 2017). The costs for storage in digital devices get less expensive as digital space becomes widely used (Amora, 2018). Both individuals and corporations can own large amounts of storage and store as many records as they want. For example, in digital forensic investigations, there can be cases where multiple copies of a revised contract appear within a system in several locations across different drives. Digital forensics investigators may take a long time to analyze these drives due to the large amounts of data stored on the drives. Another challenge in digital forensics investigations is the increased complexity of technology,

which causes heavier workloads due to the time needed to accurately analyze and reconstruct evidence (Caviglione et al., 2017).

Fuzzy logic is a technique used for representing and manipulating uncertain information (Hannan et al., 2019). Fuzzy logic helps with human reasoning and decision-making processes involving inaccurate information, estimation, inaccuracy, or non-statistical sources (Jung-Sun Kim et al., 2004; Sobrinho & Junior, 2020). Implementing fuzzy logic will help analyze forensic disk images for text files with any similarities. The proposed algorithm will use fuzzy logic to analyze the file systems quickly so investigators can save time while analyzing systems.

1.2 Research Question

This research aims to improve the analysis phase of the digital forensics investigation by verifying the reliability of fuzzy logic in identifying text files with similarities. This research will use a proposed algorithm to evaluate the reliability of fuzzy logic to determine if there are any similar text files within a forensic disk image by using fuzzy hashes. Thus, the research question that this study will be answering is: "Can the use of fuzzy logic reliably identify files with similarities on a disk image?"

1.3 Outline

The following chapters of this paper are ordered accordingly: Chapter II is a literature review of all materials relevant to this study. Chapter III reviews the research model and hypotheses that the study will follow. Chapter IV displays the results of the experiment. Finally, Chapter V analyzes the experiment results and suggests future work.

CHAPTER II

LITERATURE REVIEW

2.1 Digital Forensics Investigation and Imaging

The first phase in a digital forensics investigation is the collection of the digital device(s). In this phase, the investigator acquires the digital devices for the case. Acquisition in digital forensic investigations is the process of imaging and validating digital devices (Hou et al., 2020). The investigator then records the conditions and whereabouts of the digital devices at the scene. (Varol & Sönmez, 2017; Yankson & Davis, 2019). Next, the investigator creates a report with the acquired digital device(s) information(Varol & Sönmez, 2017). Finally, investigators make a copy of the original digital device by creating a forensics image (Khalaf & Varol, 2019).

A forensic image is a bit-by-bit copy of the original evidence that contains all the information and data found on the actual digital device (Ali et al., 2005; Yang et al., 2020). The imaging process is vital during the analysis phase because it allows forensic investigators to freely examine and analyze the forensic image without accidentally tampering with the original digital device (Varol & Sönmez, 2017). For example, suppose an investigator performs a digital forensic analysis on the actual digital device. In that case, there is no way for another expert to verify the validity of the analysis if the original digital device is not protected or encapsulated by a forensic image (Schneider et

al., 2020). However, with the actual digital device and the forensic disk image, experts will be able to compare both and verify the authenticity of the information being presented by checking the hashes of each.

The second phase in conducting a digital forensics investigation is the examination of the digital device(s). Before the analysis of the acquired evidence can be performed, it is required to check the digital device drive for any hidden, deleted, or transfigured data by using forensic tools and techniques (Abdullah et al., 2020; Varol & Sönmez, 2017). During this phase, the investigator will collect any data from the image valuable to the case. Once all relevant evidence is collected, the investigator will move on to the third phase of the investigation.

The third phase in conducting a forensics investigation is the analysis of the evidence. The purpose of this phase is to analyze the acquired digital device by using forensic image copies of the original system by using specified hardware and software tools (Abdullah et al., 2020; Varol & Sönmez, 2017). The analysis phase creates a basis for the trial and helps identify a possible conclusion for the judicial authority based on the data found on the digital device (Varol & Sönmez, 2017; Yankson & Davis, 2019). The forensic investigator will need to identify the type of crime committed, how the crime was committed, and the file locations of the evidence on the digital device (Abdullah et al., 2020; Varol & Sönmez, 2017). Any data found related to the crime is analyzed further, such as the possible incidents that could have occurred, the time of the incidents, solid and electronic evidence, and the type of software used (Varol & Sönmez, 2017; Yankson & Davis, 2019).

The fourth phase of a digital forensics investigation is the reporting of the evidence found. The investigator records the processes when handling the evidence into a report. Methods can include how the forensic image copies were made, the types of digital devices used, the operating system the digital devices were running, and what software and tools were used by the investigator to analyze the evidence (Shrivastava et al., 2013; Varol & Sönmez, 2017). The purpose of the reporting phase is to prove the evidence has not been altered and to show the processes performed on the system can be replicated again by another investigator to obtain the same results (Varol & Sönmez, 2017; Yankson & Davis, 2019).

This study will focus on improving the analysis phase of the digital forensic investigation by providing a proposed algorithm to quickly analyze multiple forensic disk images for high similarity files with the implementation of fuzzy logic. This proposed algorithm would help identify similar files that the investigator would not have known about unless they searched for this. There may be similar files in multiple drives because there may be an instance where an individual copies a file with incriminating data onto another storage devices. An investigator may not find that incriminating file, but if an individual were to reproduce it in different locations, there might be a chance of discovering this file. Of course, the proposed algorithm may find the incriminating file with the slightest chance. Still, it also means there is a chance that the proposed algorithm locates an incriminating file, which an investigator can use in a digital forensic investigation.

2.2 Digital Storage

Digital storage is the recording or storing of information or data in a storage medium. There are two types of digital storage: dynamic and static. Dynamic storage allows the user to work within the storage medium, change, and save to a file. Static storage enables users to preserve records and create a non-changing version that outsiders cannot easily alter or delete. Examples of static storage are DVDs and CDs.

As the cost for additional storage in systems gets less expensive with the evolution of technology, it makes it more common to find systems with large amounts of storage (Amora, 2018). With systems expanding their storage, there will be more space for individuals and corporates to store more records on their storage devices and servers. The growth of storage for data will overcome forensics tools without the processing speed to analyze all of the information on a system (Amora, 2018). There are cases where investigators will collect large storage systems to process the system for evidence (Caviglione et al., 2017). There will be issues of acquiring, storing, and processing these systems' data for digital forensics investigators due to the large amounts of data they will have to analyze (Caviglione et al., 2017). Suppose multiple systems with ample storage store copies of files in several locations. In that case, it may take a digital forensics investigator a long time to go through these systems. It can also be hard or almost impossible to find any relationship between the file systems and their files.

A file system controls how data is stored and retrieved within a system (Khalaf & Varol, 2019). File systems organize data by giving it a file name to be easily identifiable and isolated from others. Without a file system, any digital data that someone saves would be stored in a large body of information without a way to identify when one file

ends or when the other starts (Zhang et al., 2020). Directories allow the user to group files into separate collections or folders.

2.3 Fuzzy Logic

The term "Fuzzy Logic" was first introduced by Lotfi Zadeh in 1965 (Dzitac et al., 2017; Zadeh, 1988; Zadeh & Aliev, 2018). Fuzzy logic is a technique used for representing and manipulating uncertain information (Hannan et al., 2019). In traditional logic, values are represented as true (1.0) or false (0.0). Traditional logic uses AND, OR, and NOT to return a 0 or a 1. Values represent a range between truth and false in fuzzy logic. Fuzzy logic also uses AND, OR, and NOT to return a value in an array of [0,1]. Fuzzy logic systems enable problem-solving that would be difficult or impossible to solve without the help of mathematical models. It is a technique that helps with human reasoning and decision-making processes involving inaccurate information, estimation, inaccuracy, or non-statistical sources (Jung-Sun Kim et al., 2004; Sobrinho & Junior, 2020).

To determine the fuzziness of a value, you must look at its membership function (MF) (Mondal et al., 2017; Yao et al., 2018). A membership function represents the degree of truth in the value (Mondal et al., 2017; Yao et al., 2018). A membership function also includes whether the values in the fuzzy are discrete or continuous. Fuzzy logic architecture has four main parts: rule base, fuzzification, inference engine, and defuzzification (Hannan et al., 2019; Mondal et al., 2017; Yao et al., 2018). The rule base contains all the rules and the if-then statements provided by the persons to control the decision-making process (Hannan et al., 2019; Mondal et al., 2017). Table 1 provides

visualization for the rules-based system. By looking at the table, it is shown if the error is PS and the change of error is PB; then the output is PM.

Table 1. Fuzzy Rule Table for the Rules Base System (Hannan et al., 2019)

Error → / Change of error ↓	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NM	NM	NS	NS	ZE
NM	NB	NM	NM	NS	NS	ZE	PS
NS	NM	NM	NS	NS	ZE	PS	PS
ZE	NM	NS	NS	ZE	PS	PS	PM
PS	NS	NS	ZE	PS	PS	PM	PM
PM	NS	ZE	PS	PS	PM	PM	PB
PB	ZE	PS	PS	PM	PM	PB	PB

Fuzzification provides the resources to convert the crisp values for the binary process (Hannan et al., 2019; Yao et al., 2018). Crisp values have not been assigned a truth value yet (Mondal et al., 2017). The fuzzification process takes the numbers given and converts them into fuzzy sets. Next, the Inference Engine provides the resources to determine if the degree of the fuzzy inputs and the rules match (Mondal et al., 2017; Yao et al., 2018). The match percentage will determine which rules will need to be implemented according to the given input. Once this is complete, the applied rules will be combined to develop the control actions (Hannan et al., 2019). Finally, the defuzzification converts the fuzzy sets back into a crisp value based on the selected technique best suited for the system (Hannan et al., 2019; Mondal et al., 2017). Figure 2 shows the fuzzy logic architecture.

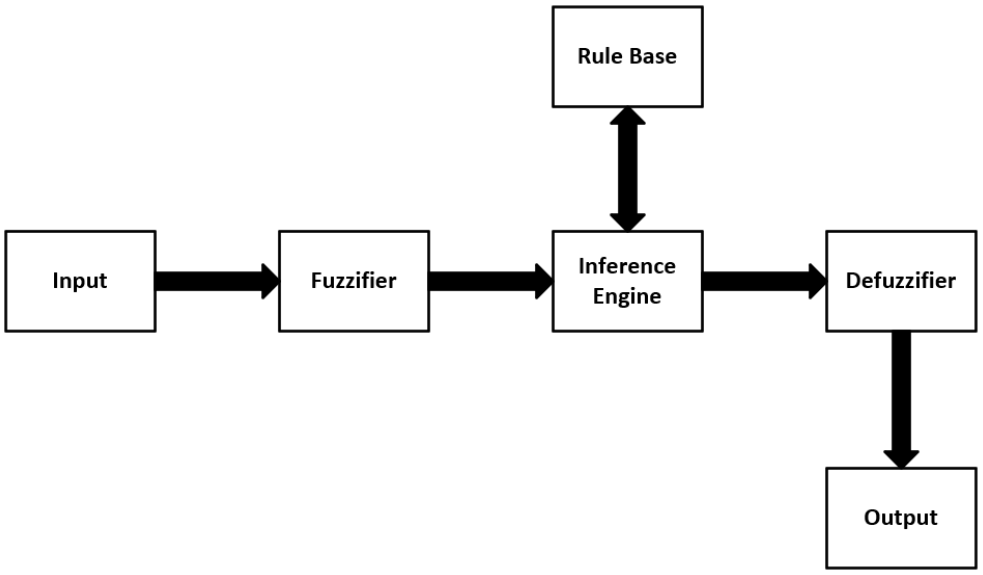

Figure 1. Fuzzy Logic Architecture (Mondal et al., 2017)

Many fuzzy logic models are available (Mittal et al., 2020). However, this paper will only cover two types of fuzzy logic models. The first is type 1 fuzzy logic which deals with nonlinear, imprecise, or missing data (Sobrinho & Junior, 2020). Type 1 fuzzy logic is the most straightforward fuzzy logic model to design and implement on older or limited hardware devices (Sobrinho & Junior, 2020). The second is type 2 fuzzy logic, an extended version of type 1 fuzzy logic (Mittal et al., 2020). Type 2 fuzzy logic differs from type 1 fuzzy logic as it can handle related linguistic problems by using account unreliability and the vagueness of information (Mittal et al., 2020). Fuzzy logic should only be used when common sense cannot solve the problem.This study utilized type 2 fuzzy logic.

2.4 Similarity Hashes

Cryptographic hashing algorithms are used to take an input and generate a fixed value called a hash that corresponds to the information (Kornblum, 2006). Hashes are used for verifying the authenticity of copied evidence. A small change in the input will drastically change the hash. Forensic investigators mainly use cryptographic hashing algorithms to create a specific output for given information (Chang et al., 2019; Kornblum, 2006). However, cryptographic hashing algorithms can only identify an exact copy of another file and do not account for similar files (Chang et al., 2019).

Approximate matching is a generic technique for finding similarities between given files by assigning a similarity score (Chang et al., 2019; Roussev, 2011). It can be categorized into three types: bytewise, syntactic, and semantic matching. Bytewise hashing, also known as fuzzy hashing or similarity hashing, measures the similarity of a given input at the byte level without analyzing the internal structure of the information (Chang et al., 2019). Syntactic matching is used to measure the similarity of a given input based on its internal structure, while semantic matching uses the given input based on its contextual attributes (Chang et al., 2019; Roussev, 2011). This study will focus on utilizing the bytewise hashing technique.

There are a few limitations and issues that occur with similarity hashing. A restriction of similarity hashing is that many fuzzy hashing methods are dependent on the block size and overall size of the targeted file for hashes (Naik et al., 2019). Limitations like this can be evaded by appending data to a file so that the header and section data are identical (Naik et al., 2019).

2.5 Fuzzy Hashing Algorithms

There are many fuzzy hashing algorithms available but the two most popular ones are SSDEEP and SDHASH. SSDEEP computes context-triggered piecewise hashes (CTPH) or fuzzy hashes. CTPH identifies the homologous sequences between unknown and known inputs/files in a specific order. SSDEEP creates many checksums by dividing the targeted file into discrete pieces and then hashing those pieces individually (Lee & Atkison, 2017). SSDEEP uses a rolling hash to compute a checksum on the last 7 bytes of the input and removes any old bytes from the end by adding new bytes to the front (Lee & Atkison, 2017). While SSDEEP uses a rolling hash, a traditional hash is computed in the background. The rolling hash produces a trigger value that the traditional hash records and concatenates to the signature and is then reset (Lee & Atkison, 2017). The file's block size determines trigger points; SSDEEP must know how large the file is to divide it into blocks. The block size can be adjusted as SSDEEP is running. For example, SSDEEP retuned a hash based on the file's initial block size and another hash based on twice the file's initial block size (Lee & Atkison, 2017). Changing a few bytes of a file will only change a small part of the overall hashes which means SSDEEP is only affected by changes locally (Lee & Atkison, 2017). The output should be about 80 bytes long. Figure 3 shows the process of generating a fuzzy hash value.

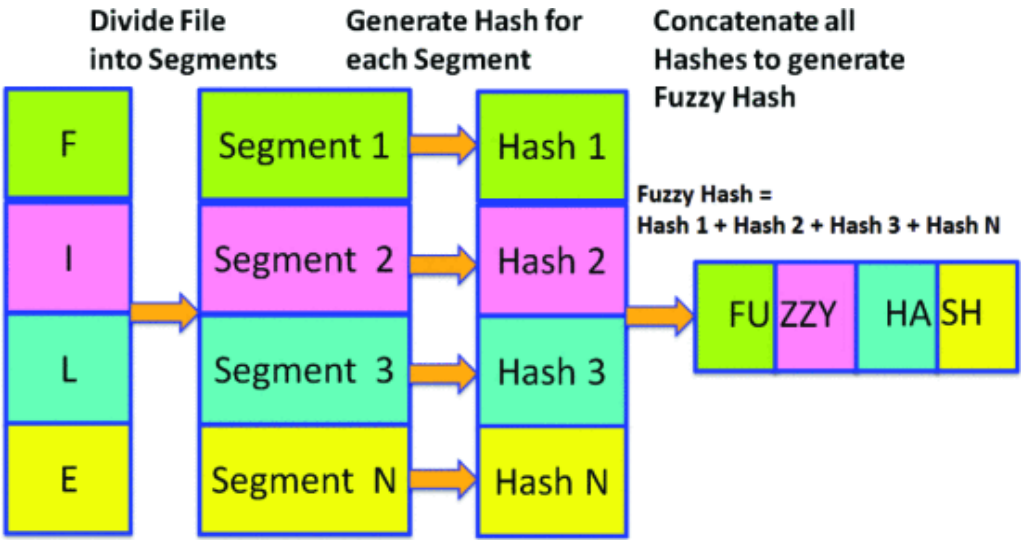

Figure 2. Generation of Fuzzy Hash Value (Naik et al., 2019)

If the output is too large or too small per recommended size, the file's block size is altered by SSDEEP, and the process is repeated; the process can become lengthy if the file is more extensive than expected. Fuzzy hashes can match inputs that have homologies. These inputs have a sequence of identical bytes in the same order (Kornblum, 2006). SSDEEP provides a library to generate and compare fuzzy hashes. Though there are other hashing programs, SSDEEP is still one of the primary choices because its speed and SSDEEP is a de facto standard (Kornblum, 2006). In addition, SSDEEP is widely used for simple identification purposes.

There are a few issues that occur with using SSDEEP. One problem is that SSDEEP is dependent on the block size and overall size of the targeted file for generating fuzzy hashes (Naik et al., 2019). In addition, an issue that occurs with the use of SSDEEP is that it cannot compare files whose sizes are less than half or more than twice the size of the file that it is being compared to (Shiel & O'Shaughnessy, 2019). Finally, another

concern with SSDEEP is that it cannot hash files over 2GB in size (Lee & Atkison, 2017).

SDHASH is a similarity hashing tool based on statistically improbable features (Naik et al., 2019). SDHASH identifies elements from the object or file that are the least likely to occur in other instances and uses the identified segment to represent the object or file (Moia & Henriques, 2017). These specified features are hashed and added to a Bloom filter known as a probabilistic set representation. When a bloom filter reaches max capacity, a new filter is created until all identified features are added (Roussev, 2011). One of the issues with SDHASH is that it takes a long time to generate the hashes (Moia & Henriques, 2017).

2.6 Next Steps

Reducing the time spent performing analysis on large amounts of data can impact digital forensic investigations by assisting investigators in analyzing data quickly. High similarity files found using fuzzy hashes may show that there may be a reason for them having similarities. Utilizing fuzzy logic to identify the relationship between these text files on the forensic disk image will assist the investigators in identifying any relationships and patterns within the forensic disk image. Identifying similar files in these forensic disk images may lead to extra data collected for evidence. Large amounts of data can be stored on a system, making it easy to overlook a file, but finding similarities of a file within the system may lead to discovering potential evidence. The chances of finding these files may be low but looking for these files may prove helpful in cases.

CHAPTER III

RESEARCH METHODOLOGY

3.1 Research Goal and Question

This study was used to determine if fuzzy logic can reliably aid in identifying similarities between text files. Utilizing fuzzy logic to help identify high similarity files in a digital storage device or disk image may assist digital forensics investigators in quickly identifying possible or crucial information that may pertain to their investigation. With the evolution of technology, the amount of stored digital data in a system will inevitably increase.

Based on the research goal, the question that this research aims to answer for this study is: “Can the use of fuzzy logic reliably identify files with similarities on a disk image?”

3.2 Hypotheses

This study determined if the use of fuzzy logic can help improve the analysis phase of the digital forensics investigation by providing an alternative way to analyze data across multiple forensic disk images with the use of fuzzy logic. There are a total of 9 hypotheses for this study. The first 8 hypothesis will determine if fuzzy logic was reliable in identifying a similarity score with a 10% difference at each individual

granularity. The ninth hypotheses will determine overall if fuzzy logic should be used to identify files if the level of similarity is unknown. Based on the acceptance of the prior hypotheses, the ninth hypotheses will be accepted if all prior hypotheses are accepted. The hypotheses are listed as follows:

Group A: 20% granularity

H_1: Fuzzy logic will identify a similarity score within a 10% difference at the 20% granularity level.

Group B: 30% granularity

H_2: Fuzzy logic will identify a similarity score within a 10% difference at the 30% granularity level.

Group C: 40% granularity

H_3: Fuzzy logic will identify a similarity score within a 10% difference at the 40% granularity level

Group D: 50% granularity

H_4: Fuzzy logic will identify a similarity score within a 10% difference at the 50% granularity level.

Group E: 60% granularity

H_5: Fuzzy logic will identify a similarity score within a 10% difference at the 60% granularity level.

Group F: 70% granularity

H_6: Fuzzy logic will identify a similarity score within a 10% difference at the 70% granularity level.

Group G: 80% granularity

H_7: Fuzzy logic will identify a similarity score within a 10% difference at the 80% granularity level.

Group H: 90% granularity

H_8: Fuzzy logic will identify a similarity score within a 10% difference at the 90% granularity level

Overall:

H_9: Fuzzy logic is reliable in identifying files with similarities across multiple forensics images when levels of similarity are unknown.

3.3 Research Method and Approach

The researcher will use a quantitative method with an experimental approach for this study. The focus of this study is to retrieve the similarity scores of the text files from a file system using a collection of scripts and analyze the scores to test the reliability of fuzzy logic in identifying files with similarities. A script will create the collection of files being used in this study to test the accuracy of fuzzy logic in identifying file similarities. In addition, this study will be using SSDEEP to create fuzzy hashes.

The independent variable in this experiment will be the collection of files (900 text files total). A bash script will randomly generate these files. The dependent variable in this experiment will be the similarity scores generated by SSDEEP

3.4 Experiment Design

The study will be using Ubuntu 20.04 LTS as the operating system on the virtual machine and will include two dynamically allocated hard disks. The first hard disk

(/dev/sda) will contain the Ubuntu operating system, the text file collection, the scripts, the experiment results directories, have a storage capacity of 50 GB, and be formatted with an NTFS file system. The second hard disk (/dev/sdb) will be used to create the disk images for experimentation, have a storage capacity of 1 GB, and will be formatted with an NTFS file system. There will be four unique disk images generated from the experiments. The steps for the experiment are as follows:

1. *generate_files.sh* is used to generate 100 sets (1 set is made up of a base file and 8 different sliced copies of the base file) of random text files (900 text files total) with fixed, known similarity scores and stored on the first drive (/dev/sda).
2. *main1.sh* has four parts:
    a. Prep the second drive (/dev/sdb) for the files to be evaluated.
    b. Mount the drive and copy 50 random sets of text files (450 text files total including the base file and its copies) from the collection of text files from the first drive (/dev/sda) to the second drive (/dev/sdb).
    c. Unmount the second drive (/dev/sdb) and create an image of drive.
    d. Make hashes of the drive and the image and save the hashes to a text file.
3. *main2.sh* has 4 parts:
    a. Mount the image as readonly.
    b. Create the ssdeep database (stored locally) from the base files on the image.
    c. Compare the ssdeep database to the sliced files on the image and stores the results in the appropriate directory (stored locally).
    d. Umount the drive.
4. Repeat steps 3 more times starting at step 2.

After the runs are completed, the researcher will extrapolate the data, which includes the similarity scores and text file names and organize the data into an excel spreadsheet.

3.5 The Proposed Algorithms

The purpose of this study is to determine if fuzzy logic can reliably aid in the identification of similarities between text files. A collection of scripts will be used to test the reliability of fuzzy logic in identifying files with similarities. Therefore, it is essential to have unique files in a dedicated drive prepared for the analysis. The experiment will utilize a total of 3 bash scripts: *generate_files.sh*, *main1.sh*, and *main2.sh*.

The *generate_files.sh* script will generate the collection of text files. This script will create 100 sets of different text files containing unique words of random sizes obtained from an English master dictionary that is built into Linux. These text files will randomly consist of 7,500 to 15,000 words labeled as file_x.txt, where x is a number from 1 to 100. These text files will then be used as the base files for generating the test files. Each base file will have 8 text copies with different granularities of 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90% of similarity to its original file . When the base file is sliced for each granularity, Bulgarian text will be appended to the end of the file to make all text files the same size (in word count) as the base. The test files will have a unique identifier that will show the level of granularity the file has to its base file. For example, take file_1.txt. The script will slice file_1.txt 8 times with different granularities so that the files created will be labeled as file_1_20.txt, file_1_30.txt, file_1_40.txt, file_1_50.txt, file_1_60.txt, file_1_70.txt, file_1_80.txt, file_1_90.txt. This file collection

will have a total of 900 text files (including the 100 base files plus their 8 sliced copies) and be stored on the first drive (/dev/sda).

The *main1.sh* script consists of three steps. The first step is cleaning and partitioning the second drive (/dev/sdb) for the experiment runs. The second step is mounting the second drive and copying 50 random set of files along with their sliced copies (450 text files total) to the mounted drive. The third step is unmounting the drive, creating an image of it, and creating the hashes for the drive and the image. The hashes will be stored in a text file locally.

The *main2.sh* script consists of 4 steps. The first step is mounting the image back as read-only. The second step is creating a ssdeep database locally from the base files on the image. The third step is to compare the sliced files on the image to the ssdeep database that is stored locally. When the fuzzy hashes are being compared, the name of the text files and their' fuzzy hash will be recorded in a log file along with a similarity score. The results of the comparison will be saved locally. The third step is to unmount the image. The proposed algorithms are only preliminary, and there may be adjustments to the method while the experiment is being done. However, the proposed algorithms were modeled thoroughly to address any issues during the experiment.

3.6 Analysis

The analysis will begin once the proposed experiments are done executing. The researcher will analayze the data from the local drive. The data includes the text file names and the similarity scores associated with them. Once this data is retrieved, the researcher will generate descriptive statistics for the data sets.

Each granularity group will have a descriptive statistic that will display the mean, standard error, median, mode, standard deviation, sample variance, kurtosis, skewness, range, minimum, maximum, sum, count and delta percent.

The min, max, and range values will show any outliers from the data set. The min value will show what the lowest value was in the run. The max value will show what the highest value was. The range will show if the data set is consistent with its intended value. For example, if a file is supposed to have 60% granularity, but the similarity score is higher or lower than 60, the range will show this on the number line.

The mean, median, and mode will give information about the characteristics of the granularity group. The standard deviation value will indicate if the granularity group is close to the mean or if the similarity scores are spread out. If the standard deviation of the granularity group is spread out, it shows that fuzzy logic was not reliable in that instance. The standard error estimates the variability across multiple samples. The sample variance measures dispersion around the mean. Kurtosis shows if the normal distribution is pointed or flat. Skewness measures if the data is heavy or light tailed to a normal distribution and if the data is shifted to the left or right. The delta percent is a statistical measure that can show how off the data is to its intended value. The closer the delta percent is to its intended value, the more reliable the scoring is. The formula for delta percent used for the experiment was (mean – granularity level)/mean * 100. The granularity level will vary based on the current data set and can be 20, 30, 40, 50, 60, 70, 80, or 90.

The hypothesis will be accepted or rejected based on the values obtained from descriptive statistics. The value that will determine if the hypotheses was accepted was the average mean score which was evaluated with the delta percent. The average mean score

needs to be within a 10% range of the intended granularity and is determined by the delta percent. If the data set fits the specified criteria, the hypothesis for the group will be accepted.

CHAPTER IV

RESULTS

4.1 Overall Results

The literature review states that SSDEEP has a problem where it cannot identify files correctly if the similarity level is less than 50% of the original text file. Therefore, to ensure the reliability of SSDEEP, the researcher analyzed the disk images to locate the text files SSDEEP could not find and gave these files a default similarity score of 0. By not assigning the similarity score of 0 to these unidentified text files, the researcher will not gauge the reliability of SSDEEP correctly due to missing data. For all runs (1-4), SSDEEP could not identify all text files in the 20% and 30% granularity levels. Appendix A contains the data from the experiment runs. This data includes details of the text files for each run, such as text file name and the similarity score assigned. With this information in mind, the results for the experiments are as follows for each run.

4.2 Group A (20%) Results

At the 20% granularity score level, SSDEEP did not identify all text files for the following runs. Run 1 identified 38 of 50 text files. Run 2 identified 36 of 50 text files. Run 3 identified 39 of 50 text files. Run 4 identified 41 of 50 text files. The descriptive statistics for the 20% runs are shown in Table 2 and Figure 3 shows a line chart of all the

runs combined. The average mean score was 28.07 which is not within a 10% acceptable score range of 18.00 to 22.00 for files containing 20% matching data. The average delta percentage between the average mean for files containing 20% matching data was 40.33%. This show that the amount of change between the files containing 20% matching data and scores produced by SSDEEP was more than 10% with a range of 11.93 and 28.07. Based on the average analysis from the four runs for a file with 20% matching score, the hypothesis H_1: Fuzzy logic will identify a similarity score within a 10% difference at the 20% granularity level is rejected.

Table 2. Descriptive Statistics for 20% Groups Runs

Statistics	R1	R2	R3	R4	Avg
Mean	28.22	25.12	28.04	30.88	28.07
Standard Error	2.39	2.30	2.22	2.20	2.28
Median	35.00	33.00	33.00	35.00	34.00
Mode	0.00	0.00	0.00	0.00	0.00
Standard Deviation	16.91	16.24	15.71	15.55	16.10
Sample Variance	286.01	263.82	246.73	241.78	259.59
Kurtosis	-0.63	-1.08	-0.37	0.39	-0.42
Skewness	-0.86	-0.83	-1.09	-1.23	-1.00
Range	58.00	47.00	47.00	58.00	52.50
Minimum	0.00	0.00	0.00	0.00	0.00
Maximum	58.00	47.00	47.00	58.00	52.50
Sum	1411.00	1256.00	1402.00	1544.00	1403.25
Count	50.00	50.00	50.00	50.00	50.00
Delta Percent	41.10	25.60	40.20	54.40	40.33

Figure 3. Line Chart for 20% Groups Runs

4.3 Group B (30%) Results

At the 30% granularity score level, SSDEEP did not identify all text files for the following runs: Run 1 identified 46 of 50 text files. Run 2 identified 48 of 50 text files. Run 3 identified 49 of 50 text files. Run 4 identified 48 of 50 text files. The descriptive statistics for the 30% runs are shown in Table 3 and Figure 4 shows a line chart of all the runs combined. The average mean score was 41.10 which is not within a 10% acceptable score range of 27.00 to 33.00 for files containing 30% matching data. The average delta percentage between the average mean for files containing 30% matching data was 37.00%. This show that the amount of change between the files containing 30% matching data and scores produced by SSDEEP was more than 10% with a range of 18.90 and 41.10. Based on the average analysis from the four runs for a file with 30% matching score, the hypothesis H_2: Fuzzy logic will identify a similarity score within a 10% difference at the 30% granularity level is rejected.

Table 3. Descriptive Statistics for 30% Groups Runs

Statistics	R1	R2	R3	R4	Avg
Mean	40.16	40.90	41.68	41.66	41.10
Standard Error	1.91	1.45	1.27	1.51	1.53
Median	43.00	43.00	44.00	43.00	43.25
Mode	46.00	47.00	46.00	47.00	46.50
Standard Deviation	13.49	10.22	8.96	10.71	10.84
Sample Variance	181.97	104.50	80.22	114.64	120.33
Kurtosis	4.26	9.00	8.45	7.78	7.37
Skewness	-2.09	-2.59	-2.14	-2.39	-2.30
Range	55.00	55.00	55.00	55.00	55.00
Minimum	0.00	0.00	0.00	0.00	0.00
Maximum	55.00	55.00	55.00	55.00	55.00
Sum	2008.00	2045.00	2084.00	2083.00	2055.00
Count	50.00	50.00	50.00	50.00	50.00
Delta Percent	33.87	36.33	38.93	38.87	37.00

Figure 4. Line Chart for 30% Groups Runs

4.4 Group C (40%) Results

The descriptive stats for the 40% runs are shown in Table 4 and Figure 5 shows a line chart of all the runs combined. The average mean score was 50.08 which is not within a 10% acceptable score range of 36.00 to 44.00 for files containing 40% matching data. The average delta percentage between the average mean for files containing 40% matching data was 25.19%. This show that the amount of change between the files containing 40% matching data and scores produced by SSDEEP was more than 10% with a range of 29.92 and 50.08. Based on the average analysis from the four runs for a file with 40% matching score, the hypothesis H_3: Fuzzy logic will identify a similarity score within a 10% difference at the 40% granularity level is rejected.

Table 4. Descriptive Statistics for 40% Groups Runs

Statistics	R1	R2	R3	R4	Avg
Mean	50.26	50.08	48.58	51.38	50.08
Standard Error	1.10	1.04	0.99	1.08	1.05
Median	50.00	50.00	49.00	52.00	50.25
Mode	57.00	47.00	50.00	57.00	52.75
Standard Deviation	7.74	7.36	7.00	7.62	7.43
Sample Variance	59.95	54.16	48.98	58.08	55.29
Kurtosis	-0.28	-0.53	-0.33	-0.34	-0.37
Skewness	-0.01	-0.05	0.12	-0.17	-0.03
Range	36.00	33.00	30.00	34.00	33.25
Minimum	33.00	33.00	36.00	35.00	34.25
Maximum	69.00	66.00	66.00	69.00	67.50
Sum	2513.00	2504.00	2429.00	2569.00	2503.75
Count	50.00	50.00	50.00	50.00	50.00
Delta Percent	25.65	25.20	21.45	28.45	25.19

Figure 5. Line Chart for 40% Groups Runs

4.5 Group D (50%) Results

The descriptive statistics for the 50% runs are shown in Table 5 and Figure 6 shows a line chart of all the runs combined. The average mean score was 58.13 which is not within a 10% acceptable score range of 45.00 to 55.00 for files containing 50% matching data. The average delta percentage between the average mean for files containing 50% matching data was 16.26%. This show that the amount of change between the files containing 50% matching data and scores produced by SSDEEP was more than 10% with a range of 41.87 and 58.13. Based on the average analysis from the four runs for a file with 50% matching score, the hypothesis H_4: Fuzzy logic will identify a similarity score within a 10% difference at the 50% granularity level is rejected.

Table 5. Descriptive Statistics for 50% Groups Runs

Statistics	R1	R2	R3	R4	Avg
Mean	58.32	57.86	57.40	58.94	58.13
Standard Error	0.97	0.84	0.92	0.90	0.91
Median	58.00	57.00	58.00	58.00	57.75
Mode	63.00	57.00	58.00	57.00	58.75
Standard Deviation	6.83	5.91	6.54	6.39	6.42
Sample Variance	46.63	34.90	42.78	40.79	41.27
Kurtosis	0.40	0.83	0.72	1.01	0.74
Skewness	-0.10	-0.28	-0.67	-0.07	-0.28
Range	34.00	29.00	29.00	34.00	31.50
Minimum	43.00	40.00	40.00	43.00	41.50
Maximum	77.00	69.00	69.00	77.00	73.00
Sum	2916.00	2893.00	2870.00	2947.00	2906.50
Count	50.00	50.00	50.00	50.00	50.00
Delta Percent	16.64	15.72	14.80	17.88	16.26

Figure 6. Line Chart for 50% Groups Runs

4.6 Group E (60%) Results

The descriptive statistics for the 60% runs are shown in Table 6 and Figure 7 shows a line chart of all the runs combined. The average mean score was 66.37 which is not within a 10% acceptable score range of 54.00 to 66.00 for files containing 60% matching data. The average delta percentage between the average mean for files containing 60% matching data was 10.61%. This show that the amount of change between the files containing 60% matching data and scores produced by SSDEEP was more than 10% with a range of 53.63 and 66.37. Based on the average analysis from the four runs for a file with 60% matching score, the hypothesis H_5: Fuzzy logic will identify a similarity score within a 10% difference at the 60% granularity level is rejected.

Table 6. Descriptive Statistics for 60% Groups Runs

Statistics	R1	R2	R3	R4	Avg
Mean	66.54	66.26	66.00	66.66	66.37
Standard Error	0.76	0.71	0.69	0.68	0.71
Median	66.00	66.00	66.00	66.00	66.00
Mode	66.00	68.00	63.00	66.00	65.75
Standard Deviation	5.35	5.05	4.91	4.80	5.03
Sample Variance	28.62	25.50	24.08	23.09	25.32
Kurtosis	0.80	1.05	-0.11	-0.26	0.37
Skewness	0.02	0.25	-0.50	-0.18	-0.11
Range	28.00	28.00	20.00	21.00	24.25
Minimum	54.00	54.00	54.00	54.00	54.00
Maximum	82.00	82.00	74.00	75.00	78.25
Sum	3327.00	3313.00	3300.00	3333.00	3318.25
Count	50.00	50.00	50.00	50.00	50.00
Delta Percent	10.90	10.43	10.00	11.10	10.61

Figure 7. Line Chart for 60% Groups Runs

4.7 Group F (70%) Results

The descriptive statistics for the 70%runs are shown in Table 7 and Figure 8 shows a line chart of all the runs combined. The average mean score was 74.58 which is within a 10% acceptable score range of 63.00 to 77.00 for files containing 70% matching data. The average delta percentage between the average mean for files containing 70% matching data was 6.54%. This show that the amount of change between the files containing 70% matching data and scores produced by SSDEEP was less than 10% with a range of 65.42 and 74.58. Based on the average analysis from the four runs for a file with 70% matching score, the hypothesis H_6: Fuzzy logic will identify a similarity score within a 10% difference at the 70% granularity level is accepted.

Table 7. Descriptive Statistics for 70% Groups Runs

Statistics	R1	R2	R3	R4	Avg
Mean	74.56	74.78	74.20	74.78	74.58
Standard Error	0.96	0.61	0.75	0.92	0.81
Median	74.00	74.00	74.00	75.00	74.25
Mode	74.00	74.00	74.00	79.00	75.25
Standard Deviation	6.76	4.33	5.29	6.52	5.73
Sample Variance	45.64	18.79	28.00	42.54	33.74
Kurtosis	1.58	0.13	-0.10	1.78	0.85
Skewness	0.45	-0.09	-0.16	0.39	0.15
Range	36.00	20.00	25.00	36.00	29.25
Minimum	61.00	63.00	61.00	61.00	61.50
Maximum	97.00	83.00	86.00	97.00	90.75
Sum	3728.00	3739.00	3710.00	3739.00	3729.00
Count	50.00	50.00	50.00	50.00	50.00
Delta Percent	6.51	6.83	6.00	6.83	6.54

Figure 8. Line Chart for 70% Groups Runs

4.8 Group G (80%) Results

The descriptive statistics for the 80% runs are shown in Table 8 and Figure 9 shows a line chart of all the runs combined. The average mean score was 82.27 which is within a 10% acceptable score range of 72.00 to 88.00 for files containing 80% matching data. The average delta percentage between the average mean for files containing 80% matching data was 2.83%. This show that the amount of change between the files containing 80% matching data and scores produced by SSDEEP was less than 10% with a range of 77.74 and 82.26. Based on the average analysis from the four runs for a file with 80% matching score, the hypothesis H_7: Fuzzy logic will identify a similarity score within a 10% difference at the 80% granularity level is accepted.

Table 8. Descriptive Statistics for 80% Groups Runs

Statistics	R1	R2	R3	R4	Avg
Mean	82.66	81.88	82.10	82.42	82.27
Standard Error	0.83	0.69	0.78	0.83	0.78
Median	83.00	83.00	82.50	83.00	82.88
Mode	83.00	83.00	83.00	83.00	83.00
Standard Deviation	5.87	4.88	5.51	5.87	5.53
Sample Variance	34.43	23.86	30.34	34.45	30.77
Kurtosis	1.45	1.25	0.73	0.85	1.07
Skewness	0.12	0.08	0.43	0.35	0.25
Range	29.00	26.00	26.00	26.00	26.75
Minimum	68.00	71.00	71.00	71.00	70.25
Maximum	97.00	97.00	97.00	97.00	97.00
Sum	4133.00	4094.00	4105.00	4121.00	4113.25
Count	50.00	50.00	50.00	50.00	50.00
Delta Percent	3.33	2.35	2.62	3.03	2.83

Figure 9. Line Chart for 80% Groups Runs

4.9 Group H (90%) Results

The descriptive statistics for the 90% runs are shown in Table 9 and . Figure 10 shows a line chart of all the runs combined. The average mean score was 91.02 which is within a 10% acceptable score range of 81.00 to 99.00 for files containing 90% matching data. The average delta percentage between the average mean for files containing 90% matching data was 1.13%. This show that the amount of change between the files containing 90% matching data and scores produced by SSDEEP was less than 10% with a range of 88.98 and 91.02. Based on the average analysis from the four runs for a file with 90% matching score, the hypothesis H_8: Fuzzy logic will identify a similarity score within a 10% difference at the 90% granularity level is accepted.

Table 9. Descriptive Statistics for 90% Groups Runs

Statistics	R1	R2	R3	R4	Avg
Mean	91.02	91.12	90.74	91.18	91.02
Standard Error	0.49	0.52	0.52	0.51	0.51
Median	90.00	91.00	91.00	90.50	90.63
Mode	90.00	93.00	90.00	90.00	90.75
Standard Deviation	3.43	3.68	3.68	3.60	3.60
Sample Variance	11.78	13.58	13.54	12.93	12.96
Kurtosis	-0.46	-0.69	-0.60	-0.92	-0.67
Skewness	0.26	-0.24	-0.05	0.04	0.00
Range	12.00	14.00	14.00	12.00	13.00
Minimum	85.00	83.00	83.00	85.00	84.00
Maximum	97.00	97.00	97.00	97.00	97.00
Sum	4551.00	4556.00	4537.00	4559.00	4550.75
Count	50.00	50.00	50.00	50.00	50.00
Delta Percent	1.13	1.24	0.82	1.31	1.13

Figure 10. Line Chart for 90% Groups Runs

4.10 Overall Results

The hypotheses for the 20% to 60% granularity levels were rejected, while the hypotheses for the 70% to 90% granularity levels were accepted. Based on these results, hypothesis H_9: Fuzzy logic is reliable in identifying files with similarities across multiple forensics images when levels of similarity are unknown is rejected. Since the hypotheses were not accepted at all granularity levels, the overall hypothesis is rejected. The hypotheses for all granularity levels need to be accepted for the overall hypotheses to be accepted. Table 10 below shows the acceptance of all hypotheses.

Table 10. Hypotheses Results

Hypothesis	R1	R2	R3	R4
H_1	R	R	R	R
H_2	R	R	R	R
H_3	R	R	R	R
H_4	R	R	R	R
H_5	R	R	R	R
H_6	A	A	A	A
H_7	A	A	A	A
H_8	A	A	A	A
H_9	R	R	R	R

Note. R = Reject, A = Accept

CHAPTER V

CONCLUSIONS

5.1 Analysis

Based on the results, the researcher has made the following conclusions about the study. First, as the granularity level increased, the range values decreased. Second, as the granularity level increased, the mean, standard deviation, and delta percent values decreased. The rest of the chapter analyzes the data found for each granularity level.

5.2 Overall Analysis of Groups

For groups A (20%) – E (60%), fuzzy logic did not meet the criteria set by the hypotheses and resulted in the hypotheses bening rejected for these groups. Therefore, the researcher inferred that fuzzy logic is the least reliable at the 20%, 30%, 40%, 50%, and 60% granularity levels. Therefore, using fuzzy logic at these granularity levels can cause complications in digital forensics investigations. For example, suppose a digital forensics investigator uses evidence found by fuzzy logic at lower granularity levels. The evidence can be made dismissible due to the unreliability of fuzzy logic at those granularity levels.

For groups F (70%) – H (90%), fuzzy logic did meet the criteria set byt the hypothesis and resulted in the hypotheses being accepted for these groups. Therefore, the researcher inferred that fuzzy logic is the most reliable at the 70%, 80%, and 90%

granularity levels. Again, this applies to digital forensics investigations. If fuzzy logic is reliable at these levels, an investigator could use evidence found at these granularity levels without having the evidence being dismissed as unreliable right away.

5.3 Conclusions

Using fuzzy logic to identify evidence can impact digital forensics investigations positively or negatively. For example, fuzzy logic can reduce the time spent analyzing forensic disk images and allow investigators more time to work on their other cases. However, fuzzy logic could also wrongfully convict a person in an investigation because fuzzy logic unreliably found evidence. Another problem with using fuzzy logic, in this case, is that an investigator may miss evidence from a text file that was not found at the lower granularity levels. If the text file had been unknown evidence in the first place, the investigator would not have found it by using this fuzzy logic method. This method will be most useful when investigators know what they are looking for on a disk image, such as revisions of a said contract. As a result, digital forensics investigators will have to use other methods to identify unknown evidence. Fuzzy logic cannot be accepted based on the results found in this research.

5.4 Limitations and Future Research

This experiment has three known limitations. The first limitation is that this experiment only analyzed text files. The second limitation is the use of only one fuzzy hashing library. The third limitation is that all the data created and used was linear.

While the researcher rejected the overall hypotheses, many improvements can be made for future research on the reliability of fuzzy logic. The first improvement is creating and using nonlinear text files. Not all revised text files will be altered linearly in investigations, so changing the composition of the text files will affect the results of the reliability of fuzzy logic. Another improvement that can be made is using another fuzzy hashing algorithm. SSDEEP is one of many fuzzy hashing algorithms with limitations that can affect the results of the reliability of fuzzy logic. Finally, another improvement that can be made is using other formats besides text files such as JPEGS.

REFERENCES

Abdullah, H. I. M., Mustaffa, M. Z., Rahim, F. A., Ibrahim, Z.-A., Yusoff, Y., Yussof, S., Bakar, A. A., Ismail, R., & Ramli, R. (2020). Smart grid digital forensics investigation framework. *2020 8th International Conference on Information Technology and Multimedia (ICIMU)*, 200–205. https://doi.org/10.1109/ICIMU49871.2020.9243536

Ali, R. R., Mohamad, K. M., Jamel, S., & Khalid, S. K. A. (2005). A review of digital forensics methods for jpeg file carving. *Vol.*, *17*, 16.

Amora, P. R. P. (2018). Smartltm: Larger-than-memory database storage for hybrid database systems. http://www.repositorio.ufc.br/handle/riufc/42180

Caviglione, L., Wendzel, S., & Mazurczyk, W. (2017). The future of digital forensics: Challenges and the road ahead. *IEEE Security Privacy*, *15*(6), 12–17. https://doi.org/10.1109/MSP.2017.4251117

Chang, D., Ghosh, M., Sanadhya, S. K., Singh, M., & White, D. R. (2019). Fbhash: A new similarity hashing scheme for digital forensics. *Digital Investigation*, *29*, S113–S123. https://doi.org/10.1016/j.diin.2019.04.006

Dzitac, I., Filip, F. G., & Manolescu, M.-J. (2017). Fuzzy logic is not fuzzy: World-renowned computer scientist lotfi a. zadeh. *International Journal of Computers*

Communications & Control, *12*(6), 748–789. https://doi.org/10.15837/ijccc.2017.6.3111

Hannan, M. A., Ghani, Z. A., Hoque, M. M., Ker, P. J., Hussain, A., & Mohamed, A. (2019). Fuzzy logic inverter controller in photovoltaic applications: Issues and recommendations. *IEEE Access*, *7*, 24934–24955. https://doi.org/10.1109/ACCESS.2019.2899610

Horsman, G. (2017). Can we continue to effectively police digital crime? *Science & Justice*, *57*(6), 448–454. https://doi.org/10.1016/j.scijus.2017.06.001

Hou, J., Li, Y., Yu, J., & Shi, W. (2020). A survey on digital forensics in internet of things. *IEEE Internet of Things Journal*, *7*(1), 1–15. https://doi.org/10.1109/JIOT.2019.2940713

Jung-Sun Kim, Dong-Geun Kim, & Bong-Nam Noh. (2004). A fuzzy logic based expert system as a network forensics. *2004 IEEE International Conference on Fuzzy Systems (IEEE Cat. No.04CH37542)*, *2*, 879–884 vol.2. https://doi.org/10.1109/FUZZY.2004.1375521

Khalaf, R. S., & Varol, A. (2019). Digital forensics: Focusing on image forensics. *2019 7th International Symposium on Digital Forensics and Security (ISDFS)*, 1–5. https://doi.org/10.1109/ISDFS.2019.8757557

Kornblum, J. (2006). Identifying almost identical files using context triggered piecewise hashing. *Digital Investigation*, *3*, 91–97. https://doi.org/10.1016/j.diin.2006.06.015

Lee, A., & Atkison, T. (2017, April). A comparison of fuzzy hashes | *proceedings of the southeast conference*. https://dl-acm-org.libproxy.usouthal.edu/doi/10.1145/3077286.3077289

Mittal, K., Jain, A., Vaisla, K. S., Castillo, O., & Kacprzyk, J. (2020). A comprehensive review on type 2 fuzzy logic applications: Past, present and future. *Engineering Applications of Artificial Intelligence*, *95*, 103916. https://doi.org/10.1016/j.engappai.2020.103916

Moia, V., & Henriques, M. (2017). A comparative analysis about similarity search strategies for digital forensics investigations. *Anais de XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais*. XXXV Simpósio Brasileiro de Telecomunicações e Processamento de Sinais. https://doi.org/10.14209/sbrt.2017.115

Mondal, H. S., Hasan, M. T., Karmokar, T. K., & Sarker, S. (2017). Improving quality of service in cloud computing architecture using fuzzy logic. *2017 4th International Conference on Advances in Electrical Engineering (ICAEE)*, 149–152. https://doi.org/10.1109/ICAEE.2017.8255344

Naik, N., Jenkins, P., & Savage, N. (2019). A ransomware detection method using fuzzy hashing for mitigating the risk of occlusion of information systems. *2019 International Symposium on Systems Engineering (ISSE)*, 1–6. https://doi.org/10.1109/ISSE46696.2019.8984540

Rao, Y., Pradhan, D., Panda, T., & Rath, R. (2020). Digital crime and its impact in present society. *8*, 1–6.

Roussev, V. (2011). An evaluation of forensic similarity hashes. *Digital Investigation*, *8*, S34–S41. https://doi.org/10.1016/j.diin.2011.05.005

Schneider, J., Wolf, J., & Freiling, F. (2020). Tampering with digital evidence is hard: The case of main memory images. *Forensic Science International: Digital Investigation*, *32*, 300924. https://doi.org/10.1016/j.fsidi.2020.300924

Shiel, I., & O'Shaughnessy, S. (2019). Improving file-level fuzzy hashes for malware variant classification. *Digital Investigation*, *28*, S88–S94. https://doi.org/10.1016/j.diin.2019.01.018

Shrivastava, A. K., Payal, N., Rastogi, A., & Tiwari, A. (2013). Digital forensic investigation development model. *2013 5th International Conference and Computational Intelligence and Communication Networks*, 532–535. https://doi.org/10.1109/CICN.2013.115

Sobrinho, A. S. F., & Junior, F. G. (2020). Type-1 fuzzy logic algorithm for low-cost embedded systems. *Computers & Electrical Engineering*, *88*, 106861. https://doi.org/10.1016/j.compeleceng.2020.106861

Stratton, G., Powell, A., & Cameron, R. (2017). Crime and justice in digital society: Towards a 'digital criminology'? *International Journal for Crime, Justice and Social Democracy*, *6*(2), 17–33. http://dx.doi.org/10.5204/ijcjsd.v6i2.355

Varol, A., & Sönmez, Y. Ü. (2017). Review of evidence analysis and reporting phases in digital forensics process. *2017 International Conference on Computer Science and Engineering (UBMK)*, 923–928. https://doi.org/10.1109/UBMK.2017.8093563

Yang, Z., Chen, Z., Zhang, P., Liu, M., & Li, Q. (2020). An information intelligent search method for computer forensics based on text similarity. *Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy*, 79–83. https://doi.org/10.1145/3377644.3377659

Yankson, B., & Davis, A. (2019). Analysis of the current state of cloud forensics: The evolving nature of digital forensics. *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, 1–8. https://doi.org/10.1109/AICCSA47632.2019.9035336

Yao, D., Su, X., Liu, B., & Zeng, J. (2018). A mobile handover mechanism based on fuzzy logic and mptcp protocol under sdn architecture. *2018 18th International Symposium on Communications and Information Technologies (ISCIT)*, 141–146. https://doi.org/10.1109/ISCIT.2018.8587956

Zadeh, L. A. (1988). Fuzzy logic. *Computer*, *21*(4), 83–93. https://doi.org/10.1109/2.53

Zadeh, L. A., & Aliev, R. A. (2018). *Fuzzy logic theory and applications: Part i and part ii*. World Scientific Publishing.

Zareen, M. S., Waqar, A., & Aslam, B. (2013). Digital forensics: Latest challenges and response. *2013 2nd National Conference on Information Assurance (NCIA)*, 21–29. https://doi.org/10.1109/NCIA.2013.6725320

Zawoad, S., & Hasan, R. (2016). Trustworthy digital forensics in the cloud. *Computer*, *49*(3), 78–81. https://doi.org/10.1109/MC.2016.89

Zhang, S., Roy, R., Rumancik, L., & Wang, A.-I. A. (2020). The composite-file file system: Decoupling one-to-one mapping of files and metadata for better

performance. *ACM Transactions on Storage*, *16*(1), 5:1-5:18. https://doi.org/10.1145/3366684

APPENDICES

Appendix A: Experiment Results

Run 1 Results

file number	20%	30%	40%	50%	60%	70%	80%	90%
4	0	33	38	44	66	68	79	85
5	47	43	54	63	63	71	82	90
6	38	43	46	60	63	68	82	90
7	44	50	60	63	69	79	88	90
8	36	46	50	60	66	88	83	88
9	58	54	69	77	75	97	97	97
10	0	36	41	46	65	74	83	90
13	30	44	49	55	68	75	79	91
16	36	46	49	57	61	75	82	91
18	30	36	40	55	69	72	82	91
20	0	29	41	57	65	74	83	93
21	0	40	47	61	61	69	77	90
25	41	43	52	63	74	72	82	90
27	43	52	49	63	63	77	79	91
28	44	50	57	68	72	82	86	94
30	44	50	55	63	68	74	83	96
31	40	41	57	61	74	79	97	97
35	0	43	54	54	66	74	85	90
36	44	47	58	52	66	75	82	91
38	32	47	58	63	68	74	88	93
39	35	41	44	58	63	75	71	85

Run 1 Results (cont.)

45	30	49	65	69	82	83	85	90
46	30	33	35	47	58	71	82	88
47	46	46	52	60	72	72	85	97
48	36	54	58	65	74	74	75	91
50	0	0	33	47	63	75	83	90
55	0	43	57	61	63	82	83	88
56	0	0	46	58	61	61	72	88
57	32	29	40	43	54	61	71	90
59	40	54	54	66	72	82	97	97
61	38	49	57	57	61	63	79	88
63	33	38	52	55	71	75	82	97
65	40	36	47	57	61	68	82	88
66	30	0	46	54	69	82	88	93
69	29	44	50	60	71	77	85	88
70	33	41	57	57	68	74	86	97
71	35	47	47	57	63	74	83	90
73	36	50	46	65	66	74	88	97
75	35	55	58	63	68	75	83	91
82	0	35	41	50	71	82	90	93
83	38	49	50	54	69	79	85	88
86	0	38	52	52	65	80	86	93
88	40	41	60	69	68	79	80	94
90	27	0	43	52	58	69	85	90
91	38	46	47	61	71	68	79	85
92	35	47	47	63	68	79	83	93
94	0	46	61	65	66	66	75	86
95	35	40	57	58	69	80	83	90
98	33	38	46	58	66	68	80	93
99	0	46	41	50	54	63	68	85

Run 2 Results

file number	20%	30%	40%	50%	60%	70%	80%	90%
1	38	46	55	57	65	75	83	90
3	33	44	47	57	69	80	90	91
11	33	46	57	68	66	75	86	94
12	29	41	38	60	60	74	83	94
13	30	44	49	55	68	75	79	91
14	36	47	50	55	58	72	82	94
16	36	46	49	57	61	75	82	91
17	0	47	66	61	63	74	77	86
20	0	29	41	57	65	74	83	93
21	0	40	47	61	61	69	77	90
24	32	47	58	69	74	77	86	90
25	41	43	52	63	74	72	82	90
26	33	35	43	40	57	71	74	83
29	33	43	47	58	61	71	74	93
30	44	50	55	63	68	74	83	96
34	38	33	41	49	54	72	83	86
35	0	43	54	54	66	74	85	90
38	32	47	58	63	68	74	88	93
39	35	41	44	58	63	75	71	85
40	0	43	43	54	60	68	80	94
44	0	40	57	58	66	71	77	85
45	30	49	65	69	82	83	85	90
48	36	54	58	65	74	74	75	91
49	33	44	47	49	63	74	71	86
50	0	0	33	47	63	75	83	90
54	0	46	54	66	66	74	85	97
55	0	43	57	61	63	82	83	88
58	35	44	50	57	65	72	79	90
59	40	54	54	66	72	82	97	97
60	0	40	50	57	71	68	83	85
61	38	49	57	57	61	63	79	88
63	33	38	52	55	71	75	82	97
66	30	0	46	54	69	82	88	93
67	30	30	41	58	68	77	83	88
70	33	41	57	57	68	74	86	97
71	35	47	47	57	63	74	83	90
72	0	38	49	57	66	74	83	86
75	35	55	58	63	68	75	83	91

Run 2 Results (cont.)

77	33	38	50	57	65	82	77	86
78	0	36	41	50	65	79	85	96
79	0	33	44	50	72	77	83	93
81	36	44	58	57	69	75	82	93
82	0	35	41	50	71	82	90	93
84	40	46	58	58	60	80	79	93
85	47	47	54	60	71	79	82	97
88	40	41	60	69	68	79	80	94
89	36	35	41	54	69	71	86	94
92	35	47	47	63	68	79	83	93
93	25	38	38	55	69	68	74	88
98	33	38	46	58	66	68	80	93

Run 3 Results

file number	20%	30%	40%	50%	60%	70%	80%	90%
1	38	46	55	57	65	75	83	90
3	33	44	47	57	69	80	90	91
5	47	43	54	63	63	71	82	90
10	0	36	41	46	65	74	83	90
12	29	41	38	60	60	74	83	94
13	30	44	49	55	68	75	79	91
14	36	47	50	55	58	72	82	94
16	36	46	49	57	61	75	82	91
17	0	47	66	61	63	74	77	86
20	0	29	41	57	65	74	83	93
21	0	40	47	61	61	69	77	90
22	0	33	36	50	63	65	74	85
24	32	47	58	69	74	77	86	90
26	33	35	43	40	57	71	74	83
27	43	52	49	63	63	77	79	91
28	44	50	57	68	72	82	86	94
29	33	43	47	58	61	71	74	93
31	40	41	57	61	74	79	97	97
32	43	38	43	54	71	77	86	93
33	0	33	36	52	61	65	80	86
34	38	33	41	49	54	72	83	86
41	29	46	50	60	74	83	88	90
42	38	47	52	55	63	72	75	86
54	0	46	54	66	66	74	85	97
57	32	29	40	43	54	61	71	90
59	40	54	54	66	72	82	97	97
60	0	40	50	57	71	68	83	85
62	43	54	54	58	65	66	79	85
63	33	38	52	55	71	75	82	97
66	30	0	46	54	69	82	88	93
67	30	30	41	58	68	77	83	88
68	40	44	47	58	68	75	82	93
69	29	44	50	60	71	77	85	88
73	36	50	46	65	66	74	88	97
74	35	43	49	50	63	71	75	90
75	35	55	58	63	68	75	83	91
76	33	41	44	58	63	74	83	91
80	0	29	36	41	60	74	75	90

Run 3 Results (cont.)

81	36	44	58	57	69	75	82	93
82	0	35	41	50	71	82	90	93
83	38	49	50	54	69	79	85	88
85	47	47	54	60	71	79	82	97
87	36	49	50	58	66	72	85	91
88	40	41	60	69	68	79	80	94
91	38	46	47	61	71	68	79	85
92	35	47	47	63	68	79	83	93
93	25	38	38	55	69	68	74	88
94	0	46	61	65	66	66	75	86
97	36	46	50	60	66	86	88	90
98	33	38	46	58	66	68	80	93

Run 4 Results

file number	20%	30%	40%	50%	60%	70%	80%	90%
1	38	46	55	57	65	75	83	90
4	0	33	38	44	66	68	79	85
5	47	43	54	63	63	71	82	90
7	44	50	60	63	69	79	88	90
9	58	54	69	77	75	97	97	97
11	33	46	57	68	66	75	86	94
12	29	41	38	60	60	74	83	94
17	0	47	66	61	63	74	77	86
19	33	0	49	60	75	71	83	88
20	0	29	41	57	65	74	83	93
23	36	47	50	63	69	77	88	90
24	32	47	58	69	74	77	86	90
25	41	43	52	63	74	72	82	90
28	44	50	57	68	72	82	86	94
29	33	43	47	58	61	71	74	93
31	40	41	57	61	74	79	97	97
36	44	47	58	52	66	75	82	91
37	36	49	55	63	71	80	88	96
39	35	41	44	58	63	75	71	85
40	0	43	43	54	60	68	80	94
46	30	33	35	47	58	71	82	88
47	46	46	52	60	72	72	85	97
48	36	54	58	65	74	74	75	91
51	35	32	49	55	66	69	80	90
56	0	0	46	58	61	61	72	88
57	32	29	40	43	54	61	71	90
59	40	54	54	66	72	82	97	97
61	38	49	57	57	61	63	79	88
62	43	54	54	58	65	66	79	85
65	40	36	47	57	61	68	82	88
68	40	44	47	58	68	75	82	93
70	33	41	57	57	68	74	86	97
72	0	38	49	57	66	74	83	86
74	35	43	49	50	63	71	75	90
75	35	55	58	63	68	75	83	91
77	33	38	50	57	65	82	77	86
78	0	36	41	50	65	79	85	96
83	38	49	50	54	69	79	85	88

Run 4 Results (cont.)

84	40	46	58	58	60	80	79	93
85	47	47	54	60	71	79	82	97
86	0	38	52	52	65	80	86	93
88	40	41	60	69	68	79	80	94
89	36	35	41	54	69	71	86	94
92	35	47	47	63	68	79	83	93
93	25	38	38	55	69	68	74	88
94	0	46	61	65	66	66	75	86
95	35	40	57	58	69	80	83	90
96	38	47	61	65	72	82	86	91
97	36	46	50	60	66	86	88	90
100	35	41	49	57	63	79	86	94

Appendix B: Scripts

generate_files.sh

```
#!/bin/bash
#create directory to put created files in
mkdir /research/file_collection
#move to created directory for file creation location
cd /research/file_collection
#assign dictionaries to variables
dict="/usr/share/dict/american-english-insane"
bdict="/usr/share/dict/bulgarian"
#begin creation of files
for i in $(seq 1 100)
do
        #create base files
        shuf -n $(shuf -i 7500-15000 -n1) $dict -o file_$i.txt
        echo "file_$i.txt" >> base_file_log.txt
        lines=$(wc -l file_$i.txt | cut -d" " -f1)
        #create 20 percent files
        per20=$(bc <<< "$lines * .2 / 1")
```

```
        head -n $per20 file_$i.txt > file_${i}_20.txt
        slice=$(bc <<< "$lines - $per20")
        shuf -n $slice $bdict >> file_${i}_20.txt
        echo file_${i}_20.txt >> twenty.txt
        #create 30 percent files
        per30=$(bc <<< "$lines * .3 /1")
        head -n $per30 file_$i.txt > file_${i}_30.txt
        slice=$(bc <<< "$lines - $per30")
        shuf -n $slice $bdict >> file_${i}_30.txt
        echo file_${i}_30.txt >> thirty.txt
        #create 40 percent files
        per40=$(bc <<< "$lines * .4 /1")
        head -n $per40 file_$i.txt > file_${i}_40.txt
        slice=$(bc <<< "$lines - $per40")
        shuf -n $slice $bdict >> file_${i}_40.txt
        echo file_${i}_40.txt >> forty.txt
        #create 50 percent files
        per50=$(bc <<< "$lines * .5 / 1")
head -n $per50 file_$i.txt > file_${i}_50.txt
slice=$(bc <<< "$lines - $per50")
shuf -n $slice $bdict >> file_${i}_50.txt
        echo file_${i}_50.txt >> fifty.txt
```

```
	#create 60 percent files
	per60=$(bc <<< "$lines * .6 / 1")
head -n $per60 file_$i.txt > file_${i}_60.txt
slice=$(bc <<< "$lines - $per60")
shuf -n $slice $bdict >> file_${i}_60.txt
	echo file_${i}_60.txt >> sixty.txt
	#create 70 percent files
per70=$(bc <<< "$lines * .7 / 1")
head -n $per70 file_$i.txt > file_${i}_70.txt
slice=$(bc <<< "$lines - $per70")
shuf -n $slice $bdict >> file_${i}_70.txt
	echo file_${i}_70.txt >> seventy.txt
	#create 80 percent files
per80=$(bc <<< "$lines * .8 / 1")
head -n $per80 file_$i.txt > file_${i}_80.txt
slice=$(bc <<< "$lines - $per80")
shuf -n $slice $bdict >> file_${i}_80.txt
	echo file_${i}_80.txt >> eighty.txt
	#create 90 percent files
per90=$(bc <<< "$lines * .9 / 1")
head -n $per90 file_$i.txt > file_${i}_90.txt
slice=$(bc <<< "$lines - $per90")
shuf -n $slice $bdict >> file_${i}_90.txt
```

```
        echo file_${i}_90.txt >> ninety.txt
done
cd /research/scripts
```

main1.sh

```
#!/bin/bash

if [[ -z $1 ]];  then

echo "Please pass run#"

exit

fi

#make sure mnt is unmounted

umount /mnt

#wipe drive

dc3dd wipe=/dev/sdb

#delete and write

(echo d; echo w) | fdisk /dev/sdb

#create new partition

(echo n; echo p; echo; echo; echo t; echo 7; echo w) | fdisk /dev/sdb

#set file system to ntfs

mkntfs -f /dev/sdb1

#mount drive

mount /dev/sdb1 /mnt

#change to mnt folder

cd /mnt

#remove lost and found folder

#rm -rf lost+found

#make a directory to put randomly selected files into
```

```
mkdir selected
#change directory
cd /research/file_collection
#randomly select 50 files for the run and logs them into a text file
shuf -n 50 base_file_log.txt > selected.txt
#goes through selected.txt and extracts info needed for copying files to directory
cat selected.txt | while read line
do
        cp /research/file_collection/$line /mnt/selected
        echo grep $line | grep -o -P '(?<=file_).*(?=.txt)' >> selected_numbers.txt
done
#goes through new.txt and copies files over into selected directory
cat selected_numbers.txt | while read line
do
        cp file_${line}_20.txt /mnt/selected
        cp file_${line}_30.txt /mnt/selected
        cp file_${line}_40.txt /mnt/selected
        cp file_${line}_50.txt /mnt/selected
        cp file_${line}_60.txt /mnt/selected
        cp file_${line}_70.txt /mnt/selected
        cp file_${line}_80.txt /mnt/selected
        cp file_${line}_90.txt /mnt/selected
done
```

```
#unmount sdb

umount /mnt

#move selected.txt and new.txt according to run

mv selected.txt selected_numbers.txt /research/$1

#adjust image creation according to run

dcfldd if=/dev/sdb of=/research/$1/$1.dd

#check hashes of drive and image

md5sum /research/$1/$1.dd /dev/sdb > /research/$1/$1hash.txt
```

main2.sh

```
#!/bin/bash

if [[ -z $1 ]];  then

echo "Please pass run#"

exit

fi

cd /research/$1

#mount image according to run:

mount -o loop,ro,offset=$((2048*512)) $1.dd /mnt

#create ssdeep database - according to run

cd /research/$1

#ssdeep

rm /research/$1/selected.db

for x in $(cat /research/$1/selected_numbers.txt)

do

ssdeep -s /mnt/selected/file_$x.txt >> /research/$1/selected.db

done

#compare image files to ssdeep database

ssdeep -s -m /research/$1/selected.db /mnt/selected/* >> /research/$1/fuzzylog.txt

#move to file_collection to move files over

cd /research/file_collection

#copy files to /research/run1/ for file manipulaiton
```

```
cp twenty.txt thirty.txt forty.txt fifty.txt sixty.txt seventy.txt eighty.txt ninety.txt
/research/$1
#change directory
cd /research/$1
#compares text files to pull out the selected files from the run
grep -f twenty.txt fuzzylog.txt > files20.txt
grep -f thirty.txt fuzzylog.txt > files30.txt
grep -f forty.txt fuzzylog.txt > files40.txt
grep -f fifty.txt fuzzylog.txt > files50.txt
grep -f sixty.txt fuzzylog.txt > files60.txt
grep -f seventy.txt fuzzylog.txt > files70.txt
grep -f eighty.txt fuzzylog.txt > files80.txt
grep -f ninety.txt fuzzylog.txt > files90.txt
#unmount
umount /mnt
```

BIOGRAPHICAL SKETCH

Name of Author: Mindy M. Wongsa

Graduate and Undergraduate Schools Attended:

University of South Alabama, Mobile, Alabama

Degrees Awarded:

Bachelor of Science in Information Technology, 2019, Mobile, Alabama

Masters of Science in Computer Information Systems, 2022, Mobile, Alabama