A Digital Media Similarity Measure for Triage of Digital Forensic Evidence

Myeong Lim, James Jones

Chapter 7

A DIGITAL MEDIA SIMILARITY MEASURE FOR TRIAGE OF DIGITAL FORENSIC EVIDENCE

Myeong Lim and James Jones

Abstract As the volume of potential digital evidence increases, digital forensic practitioners are challenged to determine the best allocation of their limited resources. While automation will continue to partially mitigate this problem, the preliminary question about which media should be examined by human or machine remains largely unsolved. This chapter describes and validates a methodology for assessing digital media similarity to assist with digital media triage decisions. The application of the methodology is predicated on the idea that unexamined media is likely to be relevant or interesting to a practitioner if the media is similar to other media that were previously determined to be relevant or interesting. The methodology builds on prior work using sector hashing and the Jaccard index of similarity. These two methods are combined in a novel manner and the accuracy of the resulting methodology is demonstrated using a collection of hard drive images with known ground truth. The work goes beyond interesting file and file fragment matching. Specifically, it assesses the overall similarity of digital media to identify systems that might share applications and thus be related, even if common files of interest are encrypted, deleted or otherwise unavailable. In addition to triage decisions, digital media similarity may be used to infer links and associations between disparate entities.

Keywords: Drive similarity, link discovery, sector hashing, Jaccard index

1. Introduction

Digital forensic practitioners extract and process evidence from digital sources and media, often during the course of criminal investigations. Digital evidence is fragile and volatile, and requires the attention of a

trained specialist to ensure that content of evidentiary value can be effectively isolated and extracted in a forensically-sound manner. One of the roles of a digital forensic practitioner is to find supporting evidence by recovering data such as files, email and photographs from computer hard drives as well as from cell phones, flash drives, RAM chips and network devices. Cloud computing has expanded data storage to multiple geographically-dispersed systems such as game consoles, Internet of Things devices and embedded systems, which are also the targets of digital forensic investigations.

As more digital data is created and digital storage systems grow in size, forensic practitioners are overwhelmed by the volume of data to be analyzed and backlogs in digital forensic laboratories are common. According to an FBI Regional Computer Forensics Laboratory Program report [23], more than 15,000 digital devices and storage media were previewed and six petabytes of data were processed by the FBI in 2017 alone, and several Regional Computer Forensics Laboratories set the reduction of backlogs as an explicit goal. In 2018, the Digital Forensics Unit of the Department of Homeland Security Cyber Crime Center processed seven petabytes of data.

Digital forensic practitioners seek to prioritize the data sources to be analyzed given limited time, and human and computing resources. Manual and forensic-tool-based analyses may take many hours to complete for each data source. Even with automated tools such as EnCase [6], FTK and Autopsy, additional human review time is required before forensic analyses of drives can be conducted. Practitioners often do not have adequate information to make decisions about which media to work on first, something that can only be determined by spending valuable time and resources on each candidate source. The lack of efficient tools and knowledge about potential evidence on a device cause inefficiencies that can lead to critical deadlines being missed and delays in disseminating actionable information.

With limited time, digital forensic practitioners must pick and choose which digital media to review from among the many available, making media triage a necessity. While triage tools exist for explicit tasks such as finding substrings of interest and specific files, a general purpose triage method based on a similarity measure between arbitrary-sized content and a labeled collection of digital media images is required. For example, a hard drive image that shows high similarity to a cluster of previously-labeled drive images of interest can be prioritized for further analysis. The similarity may be used to infer relationships between entities and as the basis for examining additional media.

This chapter proposes a digital media similarity measure based on sector hashing and a variant of the Jaccard index to help address these challenges. The similarity measure enables forensic practitioners to quickly and accurately measure the similarity of unexamined digital media to other images that are known to be relevant or interesting. A similar image is more likely to contain evidence of interest and may be used to discover previously-unknown links between entities.

2. Background

The proposed method relies on a modified Jaccard index similarity measure computed over digital media sector hashes. The similarity is computed based on the sectors present in digital media after adjusting for known common sectors (e.g., operating system and low entropy sectors) and weighting based on sector frequency. A sector size of 512 bytes is used regardless of the actual sector size so that fragments can match across devices with different sector sizes; these are assumed to be 512 bytes or 4,096 bytes in most cases. It is also assumed that files are stored on sector (cluster) boundaries, which is generally accepted to be true. Of course, sector hashing and other content-specific techniques, including the proposed methodology, cannot be used to match identical data that is encrypted with different keys.

Cryptographic hashing computes a fixed size output for an arbitrary length input. Changes in input have unpredictable and equally significant effects on the output regardless of the scope and nature of the input changes.

Sector hashing computes the hashes of data stored in digital media sectors. File hashing computes the hashes of data stored in digital media files. Advantages of sector hashing over file hashing are that sector hashing does not require filesystem interpretation and the entire file to be present for the presence of common data to be inferred. In general, finding more content in a file increases the likelihood that the entire file is present. This facilitates analysis, but recovering entire files is not necessary to conduct triage and useful analyses. In this work, matching sectors may be the result of deleted and partially-overwritten data as well as other activity (e.g., temporary files and swap space files).

The Jaccard index (JI) is a simple and widely used similarity measure that is applied to arbitrary sets of data [24]. The index, which measures the similarity between finite sample sets, is computed as follows:

$$\mathrm{JI}(\mathrm{A},\,\mathrm{B}) = \frac{|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A} \cup \mathrm{B}|} = \frac{|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A}| + |\mathrm{B}| - |\mathrm{A} \cap \mathrm{B}|} \quad 0 \leq \mathrm{JI}(\mathrm{A},\,\mathrm{B}) \leq 1$$

where JI(A, B) is the cardinality of the intersection of sets A and B divided by the cardinality of the union of sets A and B. In this work, the sets A and B comprise sector hashes from digital media of potential interest.

3. Use Cases

The following two use cases are considered in this research:

- I have a collection of digital media. Are any of these items similar to media I have seen before and about which I care (or not care)?
- I have a collection of digital media and I know where they came from. Are any of these items similar to each other? Can their similarity tell me anything about connections between people and/or devices?

Breitinger et al. [5] state that resemblance (R) and containment (C) are two common types of similarity queries, and that a similarity identification algorithm should handle one of four use cases: (i) object similarity detection (R); (ii) cross correlation (R); (iii) embedded object detection (C); and (iv) fragment detection (C). The two use cases considered in this work are similarity detection and cross correlation between hard drive images, which are routinely encountered in law enforcement and national security investigations. In both the use cases, digital forensic practitioners may or may not know the sources of the evidentiary items and their relationships to the active investigations.

4. Previous Work

Many forensic tools and algorithms use string searches as their basis. The strings may be user-specified regular expressions that match features such as email addresses, telephone numbers, social security numbers, credit card numbers, network IP addresses and other kinds of information that might correspond to pseudo-unique identifiers [12, 15, 21, 31]. Garfinkel [11] defines a pseudo-unique identifier as "an identifier that has sufficient entropy such that within a given corpus it is highly unlikely that the identifier will be repeated by chance."

Garfinkel [11] also identified an issue with typical forensic analysis – that a hard drive image does not correlate with other images. In particular, he listed three problems: (i) improper prioritization; (ii) lost opportunities for data correlation; and (iii) improper emphasis on document recovery. He attempted to address these problems via cross-drive analysis that used pseudo-unique information such as social security numbers,

credit card numbers and email addresses. In his approach, feature extractors analyzed the string files and wrote their results to feature files. The extracted features were then applied to a multi-drive corpus to identify associations between different drives.

In the case of second-order cross-drive analysis, a different question is raised: Which drives in the corpus have the largest number of features in common? To answer this question, Garfinkel [11] implemented the Multi Drive Correlator (MDC). The input to MDC is a set of drive images with a feature to be correlated and the output is a list of (feature, drive-list) tuples. The MDC program reads multiple feature files and generates a report, which shows the number of drives on which each feature was seen, the total number of times each feature was seen on the drives and the list of drives on which each feature occurred.

Beverly et al. [1] extended this work using Ethernet media access control (MAC) addresses extracted from validated IP packets. They treated the MAC addresses and drive images as nodes, and addresses on a hard drive image as links in a graph. From the partitioned graph, they were able to obtain distinct clusters in the collection of drive images.

Young et al. [32] introduced a file-agnostic approach that leverages the speed of hashing. They employed sector hashes instead of file hashes. They compared blocks (fixed-sized file fragments) against a large data set of sector hashes, and considered individual sectors and collections of contiguous sectors (blocks or clusters). Their method is based on two hypotheses:

- If a block of data from a file is distinct, then a copy of the block found on a data storage device is evidence that the file is or was present.
- If the blocks of a file are shown to be distinct with respect to a large and representative corpus, then the blocks can be treated as if they are universally distinct.

Young et al. [32] suggest that analyses of digital media would be more accurate and faster if a database of hash values computed from fixed-sized blocks of data is used. They employed large corpora such as Govdocs [13] and the NSRL RDS [18] to populate the hash value database. Three types of sectors – singleton, paired and common sectors – were analyzed to understand the root causes of non-distinct blocks. They discovered that the major reason for encountering common sectors was that the same block existed in many files due to malware code reuse and common file container formats. In order to implement a field deployment on a laptop, Young and colleagues considered sampling sectors instead of processing all the media sectors. Several database implementations were

considered and a Bloom filter front-end was ultimately implemented to speed up generic query times [3]. Young et al. analyzed several filesystems to demonstrate the generality of their approach. However, encrypted files and filesystems were found to be problematic because the same data of interest is stored differently when encrypted.

Garfinkel and McCarrin [14] have proposed hashing blocks instead of entire files; this block hashing method inspired the similarity measure methodology proposed in this chapter. Garfinkel and McCarrin also specified the HASH-SETS algorithm that identifies the existence of files and the HASH-RUN algorithm that reassembles files using a database of file block hashes. A fixed block size (e.g., 4 KiB) may present a problem due to filesystem alignment. However, this is addressed by hashing overlapping blocks with a 4 KiB sliding window over the entire drive and moving the window one sector at a time.

Taguchi [29] experimented with different sample sizes using random sampling and sector hashing for drive triage. Given a drive, the goal was to provide a practitioner with information about the utility of continuing an investigation. If a block hash value of target data is in the database, then it is very probable that the target file is on the drive. However, if no hashes are found during sampling, then a confidence level is computed that indicates the likelihood that the target data is not on the drive.

The `spamsum` program developed by Tridgell [30] performs context-triggered piecewise hashing to find updates of files. It identifies email messages that are similar to known spam. The `ssdeep` program developed by Kornblum [16], which is based on `spamsum`, computes and matches context-triggered piecewise hash values. It is more effective than `spamsum` for relatively small objects that are similar in size. However, it is vulnerable to attacks that insert trigger sequences at the beginning of files, exploiting the fact that an `ssdeep` signature value can have at most 64 characters [4].

Roussev and colleagues [25–27] have developed a similarity digest hashing method that is implemented in a program called `sdhash`. The program finds the features from a neighborhood with the lowest probability of being encountered by chance. Each selected feature, which is a 64-byte sequence, is hashed and placed in a Bloom filter. When a filter reaches full capacity, a new filter is generated. Thus, a similarity digest is a collection of a sequence of Bloom filters.

Oliver et al. [19] have proposed a locality-sensitive hashing methodology called TLSH. TLSH populates an array of bucket counts by processing an input byte sequence using a sliding window. Quartile points are computed from the array, following which the digest header and body are constructed. The digest header values are based on the quartile

points, file length and checksum. The digest body comprises a sequence of bit pairs determined by each bucket value in relation to the quartile points. A distance score is assigned between two digests; this score is a summed-up distance between the digest headers and digest bodies. The distance between two digest headers is based on file lengths and quartile ratios. The distance between two digest bodies is computed as the Hamming distance. Experiments indicate that TLSH is more robust to random adversarial manipulations than `ssdeep` and `sdhash`.

Penrose et al. [22] have used a Bloom filter for rapid contraband file detection. The Bloom filter reduces the size of the database (hashes in this case) by an order of magnitude, but incurs a small false positive rate. Penrose and colleagues subsequently implemented a larger Bloom filter for faster access, achieving 99% accuracy while scanning for contraband files in minutes using a test dataset.

Bjelland et al. [2] present three common scenarios where approximate matching can be applied: (i) search; (ii) streaming; and (iii) clustering. In a search scenario, the data space is large compared with a streaming scenario. In a clustering scenario, the input and data spaces are the same. Approximate matching is impractical for large datasets due to its high latency.

Moia and Henriques [17] have presented steps for developing new approximate matching functions. Approximate matching functions overcome the limitations of cryptographic hash functions that cannot detect non-identical, but similar, data.

The main goal of the research described in this chapter is to compute digital media image similarity measures for efficient triage. The proposed methodology does not replace approximate hashing and other methods; instead, it employs and potentially augments them. Most similarity methods operate at the file or object levels. In contrast, the proposed methodology works at the sector level, rendering it robust to deleted and partially-overwritten data. However, the proposed methodology is vulnerable to attacks that: (i) selectively delete and overwrite content that is common with another digital device; (ii) plant false fragments to mislead the algorithm and practitioners; and (iii) wipe digital media at a low level.

5. Methodology

The proposed digital media similarity measure uses sector hashes to compute a Jaccard index, but with three modifications: (i) whitelist for removing operating system and low-entropy (non-discriminatory) sectors; (ii) frequency weight that reflects content uniqueness; and (iii)

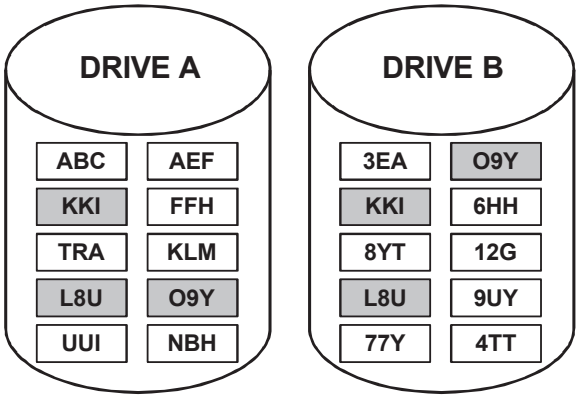

Figure 1. Two hard drives with three common sectors.

normalization that accounts for differences in media size. This section presents the Jaccard index computation over sector hashes, and the whitelist that is used to remove non-discriminatory sectors. It discusses the similarity computations for a single comparison drive and a set of comparison drives. It proceeds to show how the similarity computations are modified to account for hash frequency (i.e., sector content uniqueness). Finally, the section demonstrates the normalization of similarity computations to account for differences in digital media size.

5.1 Jaccard Index of Similarity

The basic Jaccard index is computed as the number of common sectors in two sources (e.g., hard drives A and B) divided by the number of sectors in the two sources minus the number of common sectors in the two sources. Figure 1 shows hard drives A and B, each with ten sectors. Each value denotes the hash value of the sector. The two hard drives have three of the same sectors (shaded). Thus, the Jaccard index is computed as $3/(10 + 10 - 3) = 0.1765$.

Each hard drive is divided into sectors. For example, if a hard drive has 1 terabyte (2^{40} bytes) capacity and the sector size is 512 bytes, then the drive has 2^{31} sectors. If both hard drives are exactly the same (e.g., one hard drive contains the image copied from the other hard drive), then the Jaccard index is one, indicating perfect similarity. If the two hard drives have no common sectors, then the Jaccard index is zero, indicating no similarity.

In order to compute the similarity measure, the sets of sectors in the basic Jaccard index computation are replaced by the sets of hash values

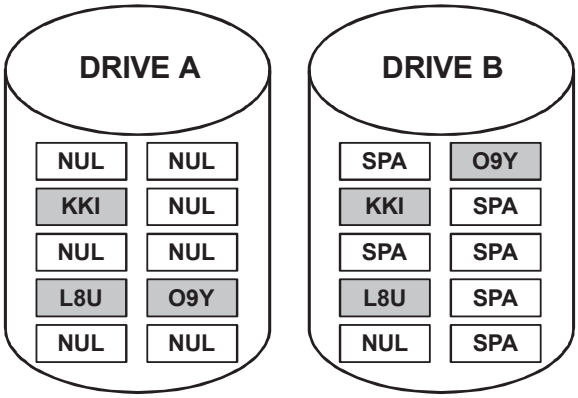

Figure 2. Whitelisted sectors (NUL and SPA).

of the sectors. The Jaccard index is thus computed as:

$$\mathrm{JI(A,\ B)} = \frac{|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A} \cup \mathrm{B}|} = \frac{|\mathrm{A} \cap \mathrm{B}|}{|\mathrm{A}| + |\mathrm{B}| - |\mathrm{A} \cap \mathrm{B}|}$$

where A and B are the sets of hash values of the sectors in the first drive and second drive, respectively.

5.2 Whitelist

Figure 2 shows that the seven non-matching sectors in drive A contain only NULL bytes and the seven non-matching sectors in drive B contain only SPACE characters. For convenience, assume that the hash value of a NULL byte sector is NUL and the hash value of a SPACE sector is SPA. If the general method for computing the Jaccard index is employed, then the index value would be the same as in Figure 1 (i.e., 0.1765).

However, the similarity measure in Figure 2 should be higher than that in Figure 1 because all the meaningful sectors match in Figure 2 (NULL and SPACE sectors are not meaningful). In fact, the similarity measure in Figure 2 should have the maximum value of one. If this situation is considered when computing the Jaccard index, a more realistic similarity measure would be obtained.

In practice, it is better to report that the two drives match exactly instead of providing a low similarity level based on the unadjusted Jaccard index of 0.1765. This is achieved by eliminating the NULL and SPACE byte sectors from the sets being compared. When these sectors are eliminated from consideration, each drive has three sectors that match exactly, yielding a Jaccard index of one. Thus, the NULL and

SPACE sectors are placed in a whitelist and all the members of the whitelist are removed before computing the Jaccard index.

In addition to the NULL and SPACE byte sectors, low entropy sectors are candidates for the whitelist. Other candidates are the sectors in a clean operating system installation. This is because, when an operating system is installed on a clean machine, sectors that are written during the installation should not contribute to the similarity measure computation.

Sector hashes corresponding to the operating systems on test drives may be saved in a database for pre-filtering (exclusion) purposes. In practice, an inventory of operating system sector hashes could be maintained and updated when new versions of the operating systems are installed. The Known File Filter (KFF) or similar tools may be used for filtering.

5.3 Similarity between Two Drives

Computing a similarity measure between two drives is straightforward, but it relies on the construction of a good whitelist. This is because the Jaccard index is computed after the sectors in the whitelist are filtered. The term "target drive" refers to the drive that is triaged using a similarity measure against a known and established "source drive."

5.4 Similarity against a Cluster of Drives

During triage, the focus is on the similarity of one target drive against multiple groups of sources that might represent different priorities, levels of interest, or specific staff members and organizations. The drives in a cluster may have been confiscated from terrorist groups or could be collections of hard drives containing malicious programs of interest. In other words, multiple different groups of source media would typically exist. After the whitelist is created and saved in a database, the sectors of the target drive and clusters of interest can be scanned and ignored based on the database. This procedure enhances the speed and accuracy of the similarity measure computations.

Given an image of interest and cluster of labeled drives, one approach to assess the image of interest is to first compute the Jaccard index values between pairs of drives in the cluster. Table 1 shows the comparison chart for a cluster of k drives. Note that JI(i, j) = JI(j, i) in the table.

The image of interest may be compared against each drive in the cluster and the Jaccard index values may be computed. Table 2 shows the corresponding comparison chart. The last row d_t may be compared against the rows d_i. Examining the values in these two sets yields a statistically meaningful assessment.

Table 1. Comparison of a cluster of drives.

	$\mathbf{d_1}$	$\mathbf{d_2}$	...	$\mathbf{d_k}$
$\mathbf{d_1}$	1	JI(1, 2)	...	JI(1, k)
$\mathbf{d_2}$	JI(2, 1)	1	...	JI(2, k)
...	...	...	1	...
$\mathbf{d_k}$	JI(k, 1)	JI(k, 2)	...	1

Table 2. Comparison of a target drive against a cluster of drives.

	$\mathbf{d_1}$	$\mathbf{d_2}$	...	$\mathbf{d_k}$
$\mathbf{d_1}$	1	JI(1, 2)	...	JI(1, k)
$\mathbf{d_2}$	JI(2, 1)	1		JI(2, k)
...	...	...	1	...
$\mathbf{d_k}$	JI(k, 1)	JI(k, 2)	...	1
$\mathbf{d_t}$	JI(t, 1)	JI(t, 2)	...	JI(t, k)

Table 3. Comparison of a target drive against an imaginary drive.

	Imaginary Drive
$\mathbf{d_1}$	JI(1, I)
$\mathbf{d_2}$	JI(2, I)
...	...
$\mathbf{d_k}$	JI(k, I)
$\mathbf{d_t}$	JI(t, I)

Another approach is to create an imaginary drive that contains all the sectors in the cluster of drives. This imaginary drive is merely the union of all the sectors in all the drives in the cluster. However, sectors that are shared by multiple drives are only counted once. The Jaccard index values are computed between the target sectors and the imaginary drive. Table 3 shows the corresponding comparison chart.

5.5 Similarity with Frequency

To simplify the analysis, an imaginary drive I is used as the source drive. The imaginary drive I is created by combining three drives C1, C2 and C3 in a cluster. Table 4 shows the sectors in the three cluster drives. Table 5 shows the imaginary drive sectors along with their frequencies.

Table 4. Sectors in cluster drives C1, C2 and C3.

C1	C2	C3
A	A	A
B	B	C
C	E	E
D	F	G

Table 5. Sectors and frequencies in imaginary drive I.

Imaginary Drive I	
A	3
B	2
C	1
D	1
E	2
F	1
G	1
Total	11

Now consider two target drives T_A and T_F, each with a single sector, A and F, respectively. Sector A in the imaginary drive I is present in every cluster drive whereas sector F is present in only one cluster drive (C2). Clearly, target drive T_A should have more similarity than T_F in a cluster comparison because sector A is present in every cluster drive. Therefore, the frequency of each sector must be used as an adjusting factor when computing the Jaccard index.

In order to illustrate the modified Jaccard index computations, two target drives T1 and T2 are compared against the imaginary drive I (and, by extension cluster drives C1, C2 and C3). Table 6 shows the sectors in the two target drives along with their frequencies in the imaginary drive.

The standard Jaccard index values for JI(T1, I) and JI(T2, I) are computed as $4/8 = 0.5$. However, T1 is more similar to I than T2 because T1 has sector A, which is shared by all three cluster drives (C1, C2 and C3), and it has sector E, which is shared by two cluster drives (C2 and C3).

Thus, the modified Jaccard index with frequency JIWF between two drives, D1 and D2, is computed as:

Table 6. Sectors and frequencies in target drives T1 and T2.

T1		**T2**	
A	3	C	1
E	2	D	1
F	1	F	1
G	1	G	1
H	1	H	1
Total	8	**Total**	5

$$\text{JIWF(D1, D2)} = \frac{\text{Nunber of Common Sectors with Frequency in D1 and D2}}{\text{Nunber of All Sectors with Frequency in D1 and D2}}$$

When computing JIWF, if a sector is shared by n drives, then the sector is counted n times. The numerator of JIWF(T1, I) is 7 because there are four common sectors (A, E, F and G) and sector A is counted three times and sector E is counted twice. The denominator of JIWF(T1, I) is 12 because there are a total of eight sectors (A, B, C, D, E, F, G and H) and A is counted three times and sectors B and E are each counted twice. Thus, JIWF(T1, I) is computed as $7/12 = 0.583$. JIWF(T2, I) is computed in a similar manner as $4/12 = 0.333$. The incorporation of sector frequencies in the JIWF computations yields a better result because T1 (0.583) is more similar to the cluster (source) drives than T2 (0.333).

5.6 Similarity with Normalized Frequency

The JIWF similarity measure is computed under the assumption that the sizes of the target and cluster drives are similar. However, if the target and cluster drives are significantly different in size – for example, the target is a thumb drive and the cluster drive is several terabytes – then, considering sector frequencies alone is inadequate when computing a good measure of similarity.

The standard Jaccard index computation employs intersection and union. The new Jaccard index computation employs modified definitions, Intersection* (I^*) and Union* (U^*), which are given by:

$$\text{Intersection}^*\text{(N1, N2)} = \text{Min}(|\text{N1}|, |\text{N2}|)$$

$$\text{Union}^*\text{(N1, N2)} = \text{Max}(|\text{N1}|, |\text{N2}|)$$

where N1 and N2 are normalized frequencies.

Table 7. Hash values and frequencies of source drive S and target drive T.

Source Drive S			Target Drive T		
Hash Value	**Frequency**	**Normalized Frequency**	**Hash Value**	**Frequency**	**Normalized Frequency**
A	5	0.3333	A	1	0.0667
B	4	0.2667	B	2	0.1333
C	3	0.2	C	3	0.2
D	2	0.1333	D	4	0.2667
E	1	0.0667	E	5	0.3333
Total	15	1	**Total**	15	1

The new Jaccard index computation employs the normalized frequency to account for the difference in the sizes of the target and cluster drives. The normalized frequency N_f is given by:

$$N_f = \frac{F_i}{S_T}$$

where F_i is the frequency of a sector hash value i and S_T is the total number of sectors in a drive.

The resulting Jaccard index with normalized frequency JINF requires two normalized values to be computed for each distinct hash value, one for the source and the other for the target. In general, the JINF similarity value is computed as:

$$\text{JINF(S, T)} = \frac{\text{Sum of all Intersection}^*\text{(S, T)}}{\text{Sum of all Union}^*\text{(S, T)}}$$

where S and T are the source and target drives, respectively.

Table 7 shows the hash values, sector frequencies and normalized sector frequencies for hypothetical source and target drives.

Table 8 shows the Intersection* and Union* values computed for the hypothetical source and target drives using the normalized frequency values in Table 7. The sum of Intersection* values over all the hashes is 0.6. Likewise, the sum of all Union* values is 1.4. The resulting JINF value is $0.6/1.4 = 0.4286$. Note that the JINF value is one for identical drives because the Intersection* and Union* values computed using the normalized frequency of each sector hash are identical.

Table 9 shows how the JINF values change when the frequency of sector hash A is successively increased by one in target drives T2, T3 and T4 (the normalized frequencies of the drives are not shown). As the frequency of the first block A in the target drive moves toward the

Table 8. Intersection* and Union* of two normalized frequency values.

Hash Value	Normalized Frequency of Source	Normalized Frequency of Target	Intersection*	Union*
A	0.3333	0.0667	0.0667	0.3333
B	0.2667	0.1333	0.1333	0.2667
C	0.2	0.2	0.2	0.2
D	0.1333	0.2667	0.1333	0.2667
E	0.0667	0.3333	0.0667	0.3333
Total			0.6	1.4
JINF			0.4286	

Table 9. JINF values of target drives T2, T3 and T4.

		T2			T3			T4	
Hash	**Freq**	**I***	**U***	**Freq**	**I***	**U***	**Freq**	**I***	**U***
A	**2**	0.125	0.3333	**3**	0.1764	0.3333	**4**	0.2222	0.3333
B	2	0.125	0.2667	2	0.1176	0.2667	2	0.1111	0.2667
C	3	0.1875	0.2	3	0.1764	0.2	3	0.1667	0.2
D	4	0.1333	0.25	4	0.1333	0.2352	4	0.1333	0.2222
E	5	0.0667	0.3125	5	0.0666	0.2941	5	0.0667	0.2778
Sum	16	0.6375	1.3625	17	0.6705	1.3294	18	0.7	1.3
JINF		0.4678			0.5044			0.5384	

frequency of the same sector hash A in the source drive, the similarity should increase. Each block is essentially a new target drive that is being checked against the source drive. For each block, the JINF similarity increases when the frequency of sector hash A increases. Note also that the total number of blocks increases by one as the frequency of sector hash A is increased by one. This increase in the total number of blocks reduces the similarity because the portion of each block against the total number of blocks decreases. In contrast, the positive effect of increasing the frequency of sector hash A is greater than the negative effect of increasing the total number of blocks.

Table 10 shows how the similarity levels increase when the frequencies of sector hash A are considered. T6 is a new target drive created from target drive T5, where the frequency of sector hash E in drive T6 is reduced by one (= 4) from 5 in drive T5. Target drive T6 has a JINF

Table 10. JINF values of target drives T5, T6 and T7.

		T5			T6			T7	
Hash	**Freq**	**I***	**U***	**Freq**	**I***	**U***	**Freq**	**I***	**U***
A	**5**	0.2631	0.3333	**5**	0.2778	0.3333	**10**	0.0667	0.3333
B	2	0.1052	0.2667	2	0.1111	0.2667	20	0.1333	0.2667
C	3	0.1578	0.2	3	0.1667	0.2	30	0.2	0.2
D	4	0.1333	0.2106	4	0.1333	0.2222	40	0.1333	0.2667
E	5	0.0667	0.2631	4	0.0667	0.2222	50	0.0667	0.3333
Sum	19	0.7263	1.2736	18	0.7556	1.2444	150	0.6	1.4
JINF		0.5702			0.6071			0.4286	

value of 0.6071, which is higher than the JINF value of 0.5702 of target T5. This is because the total number of blocks in target drive T6 is closer to the number in the source drive and has less negative impact on the JINF value computation compared with target drive T5.

Target drive T7 in Table 10 demonstrates how well the methodology copes when the target and source drives have different sizes. The frequency of each block is copied from target drive T shown in the right-hand side of Table 7 and multiplied by 10. The JINF values of target drives T and T7 are the same because the normalized frequency of each hash block is the same for both drives. Therefore, the proposed methodology does not require the sizes of the drives to be measured.

6. Validation

The proposed similarity measure was validated using the 2009 M57-Patents Scenario dataset [13], which comprises 68 hard drive images with known similarity. The images were taken from four distinct systems (named after four users, Pat, Terry, Jo and Charlie) over a 25-day period. Each system was imaged 17 times during the 25-day experiment.

For the purposes of this validation, each of the four systems represents a similar set of images. This is because they are, in fact, the same source systems with the only differences arising from normal use during the experiment. For the validation, sets of similar images were created using a subset of one user's images, following which one of the user's other images was compared with the set. High similarity was anticipated, which, in fact, occurred.

The validation was intended to serve as a preliminary confirmation that the proposed similarity measure is computationally correct, and not as a scalability test. Additional testing is planned against the Real

Data Corpus, which contains thousands of disparate media sources with no ground truth [10].

6.1 Initial Validation

The initial validation employed sequential snapshots of a single drive. On a clean drive, the following sequence of actions was performed after the Windows operating system was installed:

(a) An application was installed.

(b) An application was opened.

(c) An application was closed.

(d) An application was uninstalled.

(e) The system was rebooted.

The procedure was repeated for three clean drives with three applications, Wireshark, Firefox and Safari, whose generic drives were labeled, WS, FF and SA, respectively. A snapshot of the hard drive image were saved for each drive after each step in the sequence. In the case of drive WS, the snapshots were named WS_a, WS_b, WS_c, WS_d and WS_e. For example, WS_c denotes the snapshot of drive WS after Wireshark was closed, corresponding to action (c) in the sequence. The snapshots of the other two drives were named in a similar manner. Note that all the snapshots corresponding to a generic drive (e.g., WS) were designated as belonging to the same category, and different from the categories corresponding to the other generic drives (FF and SA).

6.2 Whitelist Sector Removal

As described above, sectors collected after the installation action (a) correspond to whitelist candidates. Therefore, sectors from snapshots WS_a, FF_a, SA_a were added to the whitelist database. The elimination of these sectors from consideration reduces the computational effort and enhances the accuracy of the Jaccard index.

6.3 JIWF Computation

The `hashdb` tool [14] was employed in this research. In order to compute the Jaccard index between two drives, A and B, the Windows operating system sectors from each drive were removed. Following this, the hash values h-A and h-B, were computed for the slimmed drives. The intersection of the hash values Int(h-A, h-B) was created using the

`hashdb` command: `intersect hash(h-A, h-B)`. Similarly, the union of the hash values Un(h-A, h-B) was created using the `hashdb` command: `add_multiple(h-A, h-B)`.

The `size` command provides the number of entries in the LMDB database [7]. In particular, it returns two values – hash data store value and hash store value. The LMDB hash store is a highly compressed optimized store of all the block hashes in the database. When scanning for a hash, if it is not in this store, then it is not in the database. Because of the degree of optimization, there can be false positives. To compensate for this, when a hash is found in the LMDB hash store, `hashdb` reads the LMDB hash data store to check that the hash actually exists. The LMDB hash data store is a multi-map store of all the hashes and their associated data and source information [8].

The hash store value was used in the Jaccard index computations. The Jaccard index JI was computed as follows:

$$\text{JI(A, B)} = \frac{\text{hash store value of Int(h-A, h-B)}}{\text{hash store value of Un(h-A, h-B)}}$$

This Jaccard index JI was used to approximate the Jaccard index with frequency JIWF.

The `add_multiple` command in `hashdb` was used to create an imaginary drive I(h-A, h-B, h-C) from hard drive images A, B and C:

$$\text{I(h-A, h-B, h-C)} = \texttt{add\_multiple(h-A, h-B, h-C)}$$

Next, the Jaccard index values for all pairs of snapshots – JI(WS_b, WS_c), JI(WS_b, WS_d), JI(WS_b, WS_e), JI(WS_c, WS_d), JI(WS_c, WS_e) and JI(WS_d, WS_e) – were computed. The Jaccard index values between WS_b and other instances became smaller after each sequence of actions. This was expected because a drive with an installed application and a drive with the same application uninstalled are less similar. After uninstallation and system rebooting, the previously matching sectors were overwritten and no longer matched. A similar trend was observed between WS_c and other instances.

The Jaccard index values of any two drives in different categories were very low compared with the Jaccard index values for any two drives in the same category.

The next set of tests employed realistic datasets, including the 2009 M57-Patents Scenario dataset [9]. The 2009 M57-Patents Scenario was created by modeling actions at a fictitious small company named M57 that was engaged in prior art searches involving patents. The dataset records actions by four employees – (i) Pat (CEO); (ii) Charlie (patent

researcher); (iii) Jo (patent researcher); and (iv) Terry (IT administrator) – over a 17-day period in November-December 2009. The hard drive in each employee's workstation was imaged daily, except for weekends and holidays. Terry's workstation ran Windows Vista Business 32-bit whereas the other three workstations ran Windows XP.

A `hashdb` instance was created for each drive. A `hashdb` instance was also created for a clean hard drive with only a Windows operating system (XP and 7). The `subtract` command in `hashdb` was used to remove Windows operating system sectors from the `hashdb` instance associated with each employee. These are referred to as "slimmed" `hashdb` instances. A imaginary drive was created by randomly choosing five random slimmed `hashdb` instances from among all the slimmed `hashdb` instances.

An imaginary `hashdb` instance `I_S_Charlie.hdb` was created from Charlie's Nov-11, Nov-20, Nov-30, Dec-04 and Dec-10 slimmed `hashdb` instances. This imaginary drive corresponded to a cluster of drives against which the similarity of a target drive would be assessed. A Jaccard index value was computed for each target drive `hashdb` instance against `I_S_Charlie.hdb`. When the target drive and cluster drives are from different categories, the similarity values would be expected to be much lower. Imaginary cluster drives were also created for the other three employees and the Jaccard index values were computed.

Figure 3 shows the JIWF results. Each sub-figure in Figure 3 shows the similarity values between the target drives of the four employees and the imaginary cluster drive created from an employee's daily images. Note that the daily images (along the x-axis) used to generate the imaginary drive associated with an employee are marked with @ symbols. Figure 3(a) shows the similarity values between the target drives of the four employees and Charlie's imaginary cluster drive. As days go by, the similarity values of Charlie's daily hard drives to Charlie's imaginary drive are higher than those of the other employees. Similar patterns are seen for Jo, Pat and Terry in Figures 3(b), 3(c) and 3(d), respectively. In the case of Figure 3(d), all the similarity values are lower compared with the similarity values in other three figures because Terry's hard drive was much larger than the hard drives of the other three employees.

6.4 JINF Computation

The JINF results shown in Figure 4 also involve the 2009 M57-Patents dataset. After creating a `hashdb` instance for each day for each employee, the `subtract` command was used to remove operating system blocks from each `hashdb` instance. An imaginary source drive was created for

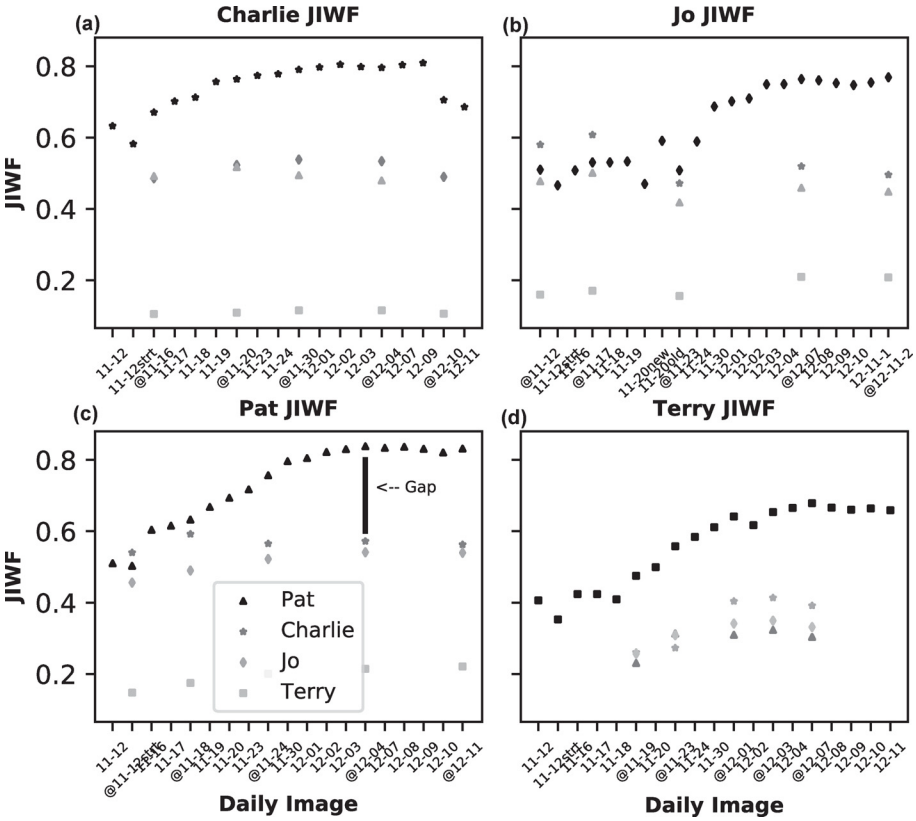

Figure 3. JIWF results for the 2009 M57-Patents Scenario dataset.

each employee by combining five random `hashdb` instances from the employee's daily drive image list. The daily images used to generate the imaginary drive associated with each employee are marked with @ symbols.

As expected, the target drives of the employees have higher similarity values when they are compared against their own imaginary source drives. In general, the similarity values of each employee's daily target drives against his/her own imaginary drive increase steadily for the first ten days, stabilize and then drop during the last few days. This is because the similarity values increase each successive day until a certain usage level is reached, after which the similarity values during successive days are about the same. The similarity values drop during the last few days because files were deleted and sector contents were overwritten, reducing the number of sectors that matched the static imaginary drives.

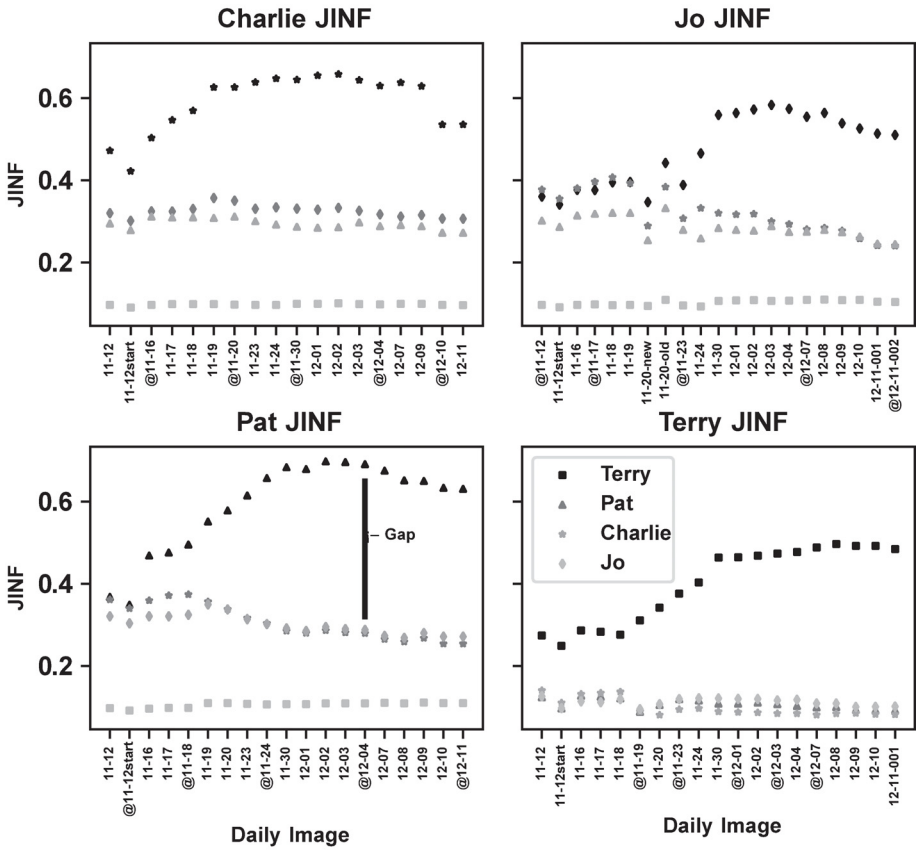

Figure 4. JINF results for the 2009 M57-Patents Scenario dataset.

In this test, a total of 305 hard drive images were compared against four imaginary drive images. Except for the first five target images belonging to Jo in Figure 4(b), all the other images yielded correct results, which is greater than 98% accuracy. The five incorrect results are for images that were taken before the images in the imaginary cluster drive; even so, they are still near the top candidates on the same day.

The experiments used a personal computer with an Intel(R) Core i7 2.30 GHz CPU and 2 TB SSD memory. It took an average of 90 minutes to compute the JINF value for a daily 40-gigabyte hard drive image.

6.5 JIWF and JINF Comparison

The JIWF and JINF methods applied to the 2009 M57-Patents Scenario dataset produce accurate results that could support triage deci-

Table 11. Comparison of JIWF and JINF performance.

	JIWF	**JINF**
Charlie	0.229789	0.265333
Jo	0.085746	0.139794
Pat	0.146044	0.279771
Terry	0.244807	0.283844
Average	0.176597	0.242186

sions. However, JINF yields larger gaps between the correct results and the next highest similarity score. Consider Pat's graphs in Figures 3 and 4. Both the graphs show similarity values between every drive in the dataset against Pat's imaginary drive. Therefore, high similarity values are expected for all of Pat's target drives. In Figure 4, where JINF values are plotted, Pat's drive on December 4 has a similarity value of 0.7 whereas Charlie's and Jo's drives have similarity values of 0.3. On the other hand, in Figure 3, where non-normalized JIWF values are plotted, Pat's drive has a similarity value of 0.8 and Charlie's and Jo's drives have similarity values of 0.55. The gap of 0.4 (= 0.7 – 0.3) obtained by the JINF method is larger than the gap of 0.25 (= 0.8 – 0.55) obtained by the non-normalized JIWF method.

Table 11 shows that JINF produces larger gaps than JIWF for all four employees. Note that the elimination of high frequency sector hashes in the JINF method yielded higher similarity values. As the cut-off value of high frequency sector hashes was gradually lowered from 5,000 to 50, the similarity values increased accordingly, which is to be expected. In other words, lowering the sector hash frequency cutoff reduces "noise" hashes and concentrates the computations on the most significant matches.

7. Conclusions

The digital media similarity measure presented in this chapter is based on a modified Jaccard index using sector hash values. The three modifications to the basic Jaccard index computation are the exclusion of a whitelist of low-entropy sectors, the incorporation of a hash frequency weight to account for content uniqueness (JIWF similarity) and the inclusion of a normalization factor to allow for accurate comparisons of media of different sizes (JINF similarity). The methodology was validated using drive images with known similarity and the highest accuracy and discrimination were obtained using the full JINF computation. The results also reveal that sector content comparisons, when appropriately

computed, can provide accurate and rapid measures of digital media similarity that support digital image triage decisions and link discovery across sources and entities.

Future research will employ a larger validation dataset to confirm the utility of the normalization factor. Additionally, it will evaluate statistical sampling instead of processing all the sectors of digital media sources to help strike the right balance between accuracy and speed. Other refinements include giving more weight to important feature sectors and considering the relative positions of matching sectors. Future work will also consider using matching sectors to direct practitioners to specific files or file remnants on digital media.

References

[1] R. Beverly, S. Garfinkel and G. Cardwell, Forensic carving of network packets and associated data structures, *Digital Investigation*, vol. 8(S), pp. S78–S89, 2011.

[2] P. Bjelland, K. Franke and A. Arnes, Practical use of approximate hash-based matching in digital investigations, *Digital Investigation*, vol. 11(S1), pp. S18–S26, 2014.

[3] B. Bloom, Space/time trade-offs in hash coding with allowable errors, *Communications of the ACM*, vol. 13, pp. 422–426, 1970.

[4] F. Breitinger and H. Baier, Performance issues about context-triggered piecewise hashing, *Proceedings of the International Conference on Digital Forensics and Cyber Crime*, pp. 141–155, 2012.

[5] F. Breitinger, B. Guttman, M. McCarrin, V. Roussev and D. White, Approximate Matching: Definition and Terminology, NIST Special Publication 800-168, National Institute of Standards and Technologies, Gaithersburg, Maryland, 2014.

[6] S. Bunting and W. Wei, *EnCase Computer Forensics: The Official EnCE: EnCase Certified Examiner Study Guide*, Wiley Publishing, Indianapolis, Indiana, 2006.

[7] H. Chu, Lightning Memory-Mapped Database Manager (LMDB), Symas Corporation, Grand Junction, Colorado (`www.lmdb.tech/doc`), 2011.

[8] Digital Corpora, `hashdb 3.1.0` Users Manual (`downloads.digitalcorpora.org/downloads/hashdb/hashdb_um.pdf`), 2017.

[9] Digital Corpora, 2009 M57-Patents Scenario (`digitalcorpora.org/corpora/scenarios/m57-patents-scenario`), 2019.

[10] Digital Corpora, Real Data Corpus (`digitalcorpora.org/corpora/disk-images/real-data-corpus`), 2019.

[11] S. Garfinkel, Forensic feature extraction and cross-drive analysis, *Digital Investigation*, vol. 3(S), pp. S71–S81, 2006.

[12] S. Garfinkel, Digital media triage with bulk data analysis and `bulk_extractor`, *Computers and Security*, vol. 32, pp. 56–72, 2013.

[13] S. Garfinkel, P. Farrell, V. Roussev and G. Dinolt, Bringing science to digital forensics with standardized forensic corpora, *Digital Investigation*, vol. 6(S), pp. S2–S11, 2009.

[14] S. Garfinkel and M. McCarrin, Hash-based carving: Searching media for complete files and file fragments with sector hashing and `hashdb`, *Digital Investigation*, vol. 14(S1), pp. S95–S105, 2015.

[15] S. Garfinkel, A. Nelson, D. White and R. Roussev, Using purpose-built functions and block hashes to enable small block and sub-file forensics, *Digital Investigation*, vol. 7(S), pp. S13–S23, 2010.

[16] J. Kornblum, Identifying almost identical files using context-triggered piecewise hashing, *Digital Investigation*, vol. 3(S), pp. 91–97, 2006.

[17] V. Moia and M. Henriques, A comparative analysis about similarity search strategies for digital forensic investigations, *Proceedings of the Thirty-Fifth Brazilian Symposium on Telecommunications and Signal Processing*, pp. 462–466, 2017.

[18] National Institute of Standards and Technology, National Software Reference Library (NSRL), Gaithersburg, Maryland (`www.nsrl.nist.gov`), 2019.

[19] J. Oliver, C. Cheng and Y. Chen, TLSH – A locality sensitive hash, *Proceedings of the Fourth Cybercrime and Trustworthy Computing Workshop*, pp. 7–13, 2013.

[20] J. Oliver, S. Forman and C. Cheng, Using randomization to attack similarity digests, *Proceedings of the International Conference on Applications and Techniques in Information Security*, pp. 199–210, 2014.

[21] H. Parsonage, Computer Forensics Case Assessment and Triage – Some Ideas for Discussion (`computerforensics.parsonage.co.uk/triage/ComputerForensicsCaseAssessmentANDTriageDiscussionPaper.pdf`), 2009.

[22] P. Penrose, W. Buchanan and R. Macfarlane, Fast contraband detection in large capacity disk drives, *Digital Investigation*, vol. 12(S1), pp. S22–S29, 2015.

[23] RCFL National Program Office, Regional Computer Forensics Laboratory Annual Report for Fiscal Year 2017, Quantico, Virginia (`www.rcfl.gov/file-repository/09-rcfl-annual-2017-190130-print-1.pdf/view`), 2017.

[24] R. Real and J. Vargas, The probability basis of Jaccard's index of similarity, *Systematic Biology*, vol. 45(30), pp. 380–385, 1996.

[25] V. Roussev, Building a better similarity trap with statistically improbable features, *Proceedings of the Forty-Second Hawaii International Conference on System Sciences*, 2009.

[26] V. Roussev, Data fingerprinting with similarity digests, in *Advances in Digital Forensics VI*, K. Chow and S. Shenoi (Eds.), Springer, Berlin Heidelberg, Germany, pp. 207–226, 2010.

[27] V. Roussev, Y. Chen, T. Bourg and G. Richard, `md5bloom`: Forensic filesystem hashing revisited, *Digital Investigation*, vol. 3(S), pp. S82–S90, 2006.

[28] W. Stallings and L. Brown, *Computer Security: Principles and Practice*, Pearson Education, Upper Saddle River, New Jersey, 2015.

[29] J. Taguchi, Optimal Sector Sampling for Drive Triage, M.S. Thesis, Department of Computer Science, Naval Postgraduate School, Monterey, California, 2013.

[30] A. Tridgell, `spamsum` (`samba.org/ftp/unpacked/junkcode/spamsum/README`), 2002.

[31] R. Walls, E. Learned-Miller and B. Levine, Forensic triage for mobile phones with DECoDE, *Proceedings of the Twentieth USENIX Security Symposium*, 2011.

[32] J. Young, K. Foster, S. Garfinkel and K. Fairbanks, Distinct sector hashes for target file detection, *IEEE Computer*, vol. 45(12), pp. 28–35, 2012.