Creating a Map of User Data in NTFS to Improve File Carving

Martin Karresand, Asalena Warnqvist, David Lindahl, Stefan Axelsson, Geir Olav Dyrkolbotn

Chapter 8

CREATING A MAP OF USER DATA IN NTFS TO IMPROVE FILE CARVING

Martin Karresand, Asalena Warnqvist, David Lindahl, Stefan Axelsson and Geir Olav Dyrkolbotn

Abstract Digital forensics and, especially, file carving are burdened by the large amounts of data that need to be processed. Attempts to solve this problem include efficient carving algorithms, parallel processing in the cloud and data reduction by filtering uninteresting files. This research addresses the problem by searching for data where it is more likely to be found. This is accomplished by creating a probability map for finding unique data at various logical block addressing positions in storage media. SHA-1 hashes of 512 B sectors are used to represent the data. The results, which are based on a collection of 30 NTFS partitions from computers running Microsoft Windows 7 and later versions, reveal that the mean probability of finding unique hash values at different logical block addressing positions vary between 12% to 41% in an NTFS partition. The probability map can be used by a forensic analyst to prioritize relevant areas in storage media without the need for a working filesystem. It can also be used to increase the efficiency of hash-based carving by dynamically changing the random sampling frequency. The approach contributes to digital forensic processes by enabling them to focus on interesting regions in storage media, increasing the probability of obtaining relevant results faster.

Keywords: File carving, hash-based carving, partition content map, NTFS

1. Introduction

The ever-increasing amounts of data handled in digital forensic investigations are a major challenge [55]. This situation has been discussed for years [8, 19, 28, 54, 58], but the challenges persist. The research community has attempted to address the problem using a number of approaches. A survey by Quick and Choo [55] lists data mining, data re-

duction and subsets, triage, intelligence analysis and digital intelligence, distributed and parallel processing, visualization, digital forensics as a service and various artificial intelligence techniques.

File carving is especially affected by the increasing amounts of data. It is used in situations where a filesystem is not present – only the properties of the stored data are available [51, 53]. Previous research has attempted to determine the data type (file type) of fragmented data using histograms of the frequencies of bytes, byte pairs and the differences between consecutive byte values [36–41]. Researchers have also leveraged the compressibility of data for type identification [2–4]. As in previous research, this work uses small blocks of data (512 B sectors) and their statistical properties to improve file carving; however, the focus is on the most probable positions of user data as opposed to their exact types.

Carving files without the help of a working filesystem is difficult, but such a capability is very valuable in digital forensic investigations. In hash-based carving, hashes of blocks of unknown data from storage media are compared against equally-sized blocks of known suspicious material. A number of strategies, techniques and algorithms for hash-based carving have been developed [6, 7, 15, 24–26, 66]. However, the large number of hash comparisons that have to be performed by a hash-based carving algorithm imposes a significant burden on the forensic process.

The research community has not as yet leveraged the principle of searching for data where it is more likely to be found. Since the allocation algorithm of an operating system places new data in a filesystem according to a set of rules (and not randomly), this principle can be used in digital forensics. However, the allocation process is too complex to understand completely and is, therefore, commonly considered to be random. As a result, the conventional approach is to search storage media from beginning to end regardless of the most probable positions of the sought-after data.

Most data of interest in forensic investigations is related to user activity (e.g., system logs and user-created files). Such data is often unique to a specific computer because the probability of two users independently creating exactly the same data is miniscule. Shared data (e.g., child abuse material) downloaded by a user is also of interest. But this data is intertwined with unique user data in storage media according to the rules of the allocation algorithm. Therefore, it is sensible to use the logical block addressing (LBA) positions of unique user data to find data related to user activities.

To demonstrate the principle, this chapter describes an experiment that uses SHA-1 hashes of the content of non-related computers running Windows 7 and later versions to compute the probabilities of unique

hashes (corresponding to unique user data) at different positions in the 30 largest NTFS-formatted partitions of 26 hard disks. The data was chosen to be as realistic as possible in order to increase the applicability of the results; for this reason, data from real-world computers was collected for the experiment.

2. Related Work

Although the approach described in this chapter is unique, it is instructive to evaluate related research in the area of file carving.

2.1 File Fragment Carving

Veenman [65] has employed the entropy, histograms and Kolmogorov complexity of 4 KiB file fragments to determine their types; the results reveal that histograms yield the highest detection rate versus false positives. Calhoun and Coles [11] have experimented with statistical measures such as ASCII code frequency, entropy, mode, mean, standard deviation and correlation between adjacent bytes; they have also considered the use of the longest common sub-strings and sub-sequences between file fragments for data classification.

Ahmed et al. [1] have used byte frequency distributions to measure the distances between the statistical properties of a data fragment and a model; instead of using the Mahalanobis distance measure, they employed cosine similarity and obtained better results. Li et al. [43] have also used byte frequency distributions (histograms) of different data fragments, but in conjunction with a support vector machine to discriminate between data types. The best results were achieved using byte frequency distributions on their own.

Fitzgerald et al. [23] have combined several statistical measures of data fragments (histograms of one- and two-byte sequences, entropy and Kolmogorov complexity) to obtain feature vectors that were fed to a support vector machine for classification. Their method outperformed the approaches proposed by other researchers. However, they did not evaluate the contributions of the chosen feature vectors, leaving it for future work. Interested readers are referred to Poisel et al. [52] for a taxonomy of data fragment classification techniques.

2.2 Hash-Based Carving

Hash-based carving compares the hashes of known file blocks against the hashes of equally-sized blocks from storage media. This approach

enables files that have been partially overwritten or damaged to be identified.

The roots of hash-based carving go back to the `spamsum` tool developed by Tridgell [62]. Garfinkel and McCarrin [25] were among the first to use hashes for file carving (specifically, in the 2006 Digital Forensic Research Workshop (DFRWS) Carving Challenge). Subsequently, Kornblum [42] employed `spamsum` for piecewise hashing, which is now referred to as approximate matching. Dandass et al. [17] have used hashes for file carving in an empirical study of disk sector hashes. However, the term hash-based carving was first introduced by Collange et al. [15] when they explored the possibility of using a graphics processing unit (GPU) to compare hashes of 512 B sections of known files with hashes of 512 B sectors from disk images.

In the 2006 DFRWS Carving Challenge [25], portions of files found on the Internet were hashed and used to find the same hashes in the challenge image. The experience led to the development of the `frag_find` tool [26]. The optimal size of the data blocks to hash was determined to be equal to the sector size; however, it is not mentioned if the sectors were 512 B or 4 KiB in size. Garfinkel et al. [25] elaborated further on the size of hashed blocks, stating that, starting with Windows NT 4.0, the default minimum allocation unit in NTFS is 4 KiB [48].

Foster [24] discusses the problem of data shared across files, noting that "the block of nulls is the most common block in [the] corpus," relating them to NULL paddings in files. Young et al. [66] have further developed Foster's ideas; they discuss the optimal block size, the handling of large amounts of data, efficient hash algorithms, good datasets to use and common blocks of files.

Random sampling has been used to improve the speed of hash-based carving [24–26]. The determination of a suitable sampling frequency is regarded as sampling without replacement. A higher sampling frequency may increase the detection rate, but it negatively impacts execution speed. The key problem is to strike a trade-off between detection rate and execution speed.

2.3 Data Persistence

The concept of data persistence is relevant to this research because persistence in areas of storage media indicates that these areas have not been reused. This information is valuable when creating a map of storage media.

Jones et al. [35] have created a framework for studying the persistence of deleted files in storage media. They employ differential forensic anal-

ysis to compare snapshots of filesystems in use and follow the decay of deleted files over time.

Fairbanks and Garfinkel [22] identify twelve factors that affect data persistence in storage media. Fairbanks [20, 21] also discusses the low-level functions of Ext4 and their impacts on digital forensics.

2.4 Data Reduction

Quick and Choo [54, 56] have proposed methods for reducing the amounts of data analyzed in digital forensic investigations. They extract specific files using a list of key files and work on the subset of files. However, this requires a working filesystem, which limits the applicability of their methods. Also, the list of key files has to be updated constantly.

Rowe [59] has proposed a similar approach to that of Quick and Choo, although it is more technical. Nine methods are compared for identifying uninteresting files, which are defined as "files whose contents do not provide forensically useful information about the users of a drive." However, all the methods require a working filesystem, which is inconsistent with the fundamental premise of file carving.

2.5 Data Mapping

Guidance Software [34] has developed an EnScript module for its EnCase software, which creates a map of the recoverable sectors of a file found in a filesystem. The module can handle situations where other tools do not work (e.g., partially damaged files). However, it is very processor intensive and can create maps of only a few files at a time.

Gladyshev and James [28] have studied file carving from a decision-theoretic point of view. They specify a model where the storage media is sampled at a frequency based on the properties of the hard drive and the file type that is to be found. In some situations, their carving model outperforms standard linear carving algorithms, but their solution is not general. Gladyshev and James also mention using the distribution of data on disk, but do not relate this to the probability of finding user data at various logical block addressing positions in storage media.

Van Baar and colleagues [63, 64] focus on the non-linear extraction of data from images. They posit that master file table (MFT) records (filesystem metadata) of an NTFS partition should be extracted first. The MFT records can be used to find interesting areas in the filesystem. They also leverage the analysis process to influence the imaging process by prioritizing certain portions of storage media during imaging.

3. Proposed Method

The review of the literature reveals that it is necessary to improve the efficiency of algorithms and tools used in digital forensics, and especially in file carving. However, the research community has not yet leveraged the inherent structures of allocation algorithms to address the problem. This research is founded on the novel idea of using the probability of finding user data in various locations in storage media to govern the digital forensic process and, thus, contribute to an immediate increase of the efficiency of current file carving algorithms and tools. In hash-based carving, the concept can be used to increase the efficiency of random sampling by varying the sampling rate according to the probability of finding user data at different logical block addressing positions. The concept can also be used during triage and other situations that involve trade-offs between speed and detection rate.

Another advantage is that the proposed method works without a filesystem. The map created by the method can be used directly to improve the speed of any file carving algorithm by revealing the most probable positions of unique data in NTFS-formatted storage media. The forensic process can be started at the position with the highest probability of containing data of interest (e.g., user data) or by varying the sampling rate according to the probability of finding user data. In the latter case, the sampling frequency would be higher where it matters most and lower in other areas, increasing the likelihood of getting hits while maintaining the same amount of samples as in the case of equally-distributed sampling.

The proposed method also benefits digital forensic analysts because the map can help plan a forensic process in the same way a road map is used to plan a trip. Storage media are currently treated as black boxes, which forces forensic analysts to spend valuable time scanning them from beginning to end before any analysis can be performed. This is especially true in a file carving situation where there is no filesystem to govern the search. With the proposed method, a forensic analyst can focus on relevant areas of the storage media and postpone, or even skip, less relevant areas.

The map can also support storage media imaging. By starting the imaging process at the most probable position of user data and continuing in decreasing order of relevance, the analysis process can be run almost in parallel with imaging because the most relevant data for analysis is available immediately. Thus, the analysis process can be started earlier, even before the imaging is finalized, saving valuable time and effort. Of course, the reliability of the analysis will increase as more

data is analyzed, but preliminary results are available that can guide the subsequent work. This concept is also supported by the Hansken Project [63, 64]. By implementing the proposed method in Hansken, its ability to handle media with broken filesystems would be enhanced, possibly approaching the performance of the standard process.

The proposed method uses 512 B sectors when hashing data in order to handle storage media regardless of its filesystem cluster size. Since the map is created once and can be reused, as in the case of a road map, no performance penalty is incurred when using the map. The map is divided into a small number of equally-sized areas (currently 128) so that any random seek penalty would only occur between – and not within – these areas, and could, therefore, be ignored.

The only situation where 512 B hashes are required is when the map is created. There is no need to use 512 B hashes when performing casework. Likewise, any hashing algorithm can be used in casework because the map hashes are only used to compute the probabilities of user data at different positions, never for comparing hashes from a specific case. If a given hash algorithm is broken, it could be replaced by a new algorithm and the map would still work.

During the research, nine sectors were discovered to have the same hash values at the same logical block addressing positions in all 30 partitions. These sectors could be used to identify an NTFS filesystem, to find the start of an NTFS partition and to locate the `$MFT` file for further processing. These tasks can be accomplished regardless of the state of the filesystem.

4. Experimental Setup

Live data was collected from real computers in order to determine the distribution of unique data in the major NTFS-formatted partitions of common Microsoft Windows computers. Next, the probabilities of finding unique hash values at different logical block addressing positions were computed. Finally, a map was created by computing the mean probabilities of a number of (currently 128) equally-sized partition areas based on logical block addressing positions. The mean probability computations were performed to generalize and scale the map into a usable format.

In order to reduce the size of the data stored in the experiment and to protect the privacy of users, the SHA-1 algorithm was used to hash each 512 B sector of all 30 NTFS-formatted main partitions considered in the experiment. SHA-1 was employed because it currently strikes the best balance between speed, collision risk and hash size from among the can-

didate hash algorithms (MD5 and SHA family). The choice was based on a practical evaluation using the available hardware. The hashing was performed locally at each source computer and only the resulting hashes were moved from the source computers.

Since the SHA-1 algorithm maps 512 B of data to a 20-byte hash, there is always a risk of collisions. The collision risk arises because 512 B of data is compressed into a 20-byte hash and, therefore, the results may contain false positives. The problem can be analyzed using the Birthday Paradox, where N is the number of possible hashes, n is the number of hashes (i.e., total number of sectors hashed in the worst case) and $Prob(\text{Collision})$ is the probability of a collision, which is given by:

$$Prob(\text{Collision}) = 1 - \frac{N!}{N^n \cdot (N-n)!}$$

When $N = 2^{160}$ and $n = 18,210,308,798$, the probability of at least one collision is:

$$Prob(\text{Collision}) \approx 1 - e^{-n^2/2N} \approx n^2/2N \approx 1.1 \cdot 10^{-28}$$

This expression uses Stirling's approximation of factorials, which yields acceptable results for large numbers.

Since the SHAttered attack [16, 61] is 100,000 times faster than a brute force attack using the Birthday Paradox, the risk of an intentional collision is higher. Fortunately, the attack is infeasible in digital forensic scenarios. Therefore, a unique SHA-1 hash is assumed to represent a unique piece of data.

Although the SHA-1 algorithm has been broken [16, 61], from a cryptographic point of view, the risk of an intentional collision is also negligible. This is because the computing power required to create a collision is out of reach of common users. Also, such an attack would require a large number of collisions to be created for the majority of the storage media in the map source data. In a digital forensic scenario, it would be much easier to fill the storage media with shared and unique data in an intentional pattern; but this is easily mitigated by collecting source data from unrelated sources. In any case, the mapping process is not limited to SHA-1 – any hashing algorithm would suffice as long as all the mapping data is hashed using the same algorithm.

4.1 Data Collection

Data representing real-world situations was obtained by using convenience sampling of data from computers owned by acquaintances. The Real Data Corpus was not used because the timestamps on the website [18] indicate that the last update to the dataset was made back in

2011. The data collected for the experiment was much newer, corresponding to Windows versions 8 and 10. Windows 8 was introduced at the end of 2012 [27] and, therefore, Windows 8 data does not exist in the Real Data Corpus, nor does Windows 10 data.

Data was collected from 30 partitions in 26 computers (23 consumer-grade and three office-grade computers). The data was collected by hashing every 512 B sector of the drives using the `dcfldd` disk imaging tool set to the SHA-1 hash algorithm. The operating system installations corresponded to three language packs ranging from Microsoft Windows 7 to Windows 10 Enterprise, Professional, Ultimate, Home and Educational versions. Some computers have been upgraded from an earlier Windows version to Windows 10. However, five computers are maintained in their original form and access to all their raw content is still available.

Real computers were employed to avoid bias introduced by simulating user behavior. While real computers ensure that the results would be as close to possible to casework, the drawback was reduced control of the material. For example, in some instances, information whether a drive was mechanical or solid-state was not available. The lack of information does not affect the results because data was collected at the logical block addressing level from the drive controller. However, the lower level physical storage formats were hidden [5, 10, 14, 57].

As far as the experiment is concerned, the only difference between a mechanical hard disk and a solid state drive is how the unused areas were filled. This could be old data, `0x00` or `0xFF` depending on how the TRIM command was implemented in the solid state drives [9, 29–33]. Hence, a mechanical drive would more often yield old data from unallocated clusters compared with a solid state drive. Since only the logical block addressing positions of unique data was used, any `0x00` and `0xFF` fillings were automatically filtered. In the case of old data from unallocated clusters, a very unbalanced erase/write cycle would leave a large amount of old data, i.e., a large amount of data would have been erased first, followed by a small amount of (or no) writing of new data. This would be the case if a hard disk was erased using a random pattern and then reformatted and reused. The experimental results would be affected if a large number of unallocated sectors were to contain old (unique) data. In order to affect the map creation process to a sufficient degree, this situation would have to hold for a significant number of partitions in the dataset. Therefore, data was not collected from the computers until their users confirmed that large-scale filesystem cleaning was not performed close to the data collection.

Table 1. Partition sizes, unique hashes, and `0x00` and `0xFF` fillings in the last 20 GB.

Partition	Size (GiB)	Unique Hashes (%)	0x00 Filling (%)	0xFF Filling (%)
F	59.5	0.08	100.00	0.00
E	59.5	22.36	2.01	0.12
AC	111.3	7.63	100.00	0.00
I	111.6	61.83	20.12	1.67
A	118.4	23.17	75.24	0.00
W	118.6	59.80	26.18	0.00
K	146.4	5.82	100.00	0.00
Qa	150.0	43.33	45.78	0.07
N	177.6	38.32	48.48	0.00
Ra	185.9	31.89	56.89	0.14
Sa	200.0	86.85	0.02	0.02
Oa	209.0	13.68	100.00	0.00
Y	217.1	14.77	100.00	0.00
P	232.7	53.97	0.83	0.07
H	237.3	16.68	100.00	0.00
AA	237.3	12.60	100.00	0.00
D	237.9	20.59	0.00	100.00
G	238.1	7.03	25.56	0.16
M	238.1	23.06	79.47	0.01
Rb	258.4	1.68	100.00	0.00
Sb	265.6	36.22	100.00	0.00
T	297.9	9.35	48.12	0.28
C	421.7	34.98	0.86	1.17
Z	423.9	4.05	100.00	0.00
X	443.8	6.48	100.00	0.00
U	448.0	10.60	100.00	0.00
V	465.6	48.43	100.00	0.00
Ob	699.0	0.15	98.72	0.00
Qb	766.5	1.47	100.00	0.00
B	905.2	29.74	100.00	0.00
Total	8,683.4	20.92	67.61	3.35

The storage media used in the experiment had capacities ranged from 64 GB to 1 TB. The largest NTFS-formatted partition was extracted from each storage media drive based on the assumption that it contained the operating system and user files. In four instances, an extra storage partition was present, which was also extracted.

Table 1 shows the partition sizes, percentages of unique hashes and the `0x00` and `0xFF` fillings in the last 20 GB of the partitions. The total size of the partitions was 8,638.4 GiB, corresponding to 18,210,308,798

hashes, 3,809,786,792 of which were unique. A low percentage of unique hashes in a partition indicates that the partition was not used or at least not used for storing user data. The larger partitions in Table 1 (i.e., those ending in "b") have low percentages of unique hashes and high percentages of `0x00` and `0xFF` fillings. Data was collected from multiple partitions on the associated drives, which were large.

The lifetimes and, hence, amounts of data stored on the drives varied. Most of the drives had `0x00` fillings to some extent. These may be remnants of the production process, but some of the smaller drives ($\leq$ 256 GiB) were solid state drives that would have been filled with `0xFF` values at the factory [9].

The last 20 B of each partition was examined to determine if any partitions in the dataset were completely filled with data at some point in their lifetimes. The 20 GB size was selected as a suitable trade-off between having a large amount of data and the risk of including operating system files in the case of the smaller partitions. Low percentages of both `0x00` and `0xFF` fillings in Table 1 is an indicator that the partition was completely filled or was wiped with a random pattern at some point in its lifetime.

NTSF-formatted partitions were considered because NTFS has about 90% of the desktop system market share [50]. The partition names in Table 1 were assigned based on the order of hashing (i.e., partition "A" was hashed before "B"). Four computers contributed two partitions each to the dataset due to the partition size (these computers were installed with an extra partition for user data). The associated partitions are indicated by a second lowercase letter in their names. Although the partitions did not have operating system files, they contained an NTFS filesystem and were, therefore, included in the dataset.

Some of the unique hash values that were found corresponded to 1 KiB MFT records. The size of an MFT record is defined in the boot sector of an NTFS partition; the standard size is 1 KiB [12]. These records yielded up to two unique hash values each due to their highly varying content (timestamps, file names, file content, etc.).

Therefore, a survey of 27 computers not included in the dataset was conducted to estimate the mean number of MFT records. This was accomplished by counting the total number of files and folders because each file and folder in a computer is represented by at least one MFT record. If a file has many attributes (e.g., alternate streams) or is heavily fragmented, then the filesystem creates a new MFT record to hold the extra information.

The survey revealed that the average total number of files and folders in the computers was 363,630. Due to the uncertainty involved in

counting (using File Explorer), an extra 25% was added to account for hidden files, files requiring more than one MFT record and MFT records internal to the filesystem. The extra 25% also covered network storage of user data in the office-grade computers. In the case of the consumer-grade computers, all the user files would most likely have been stored locally; therefore, these files were included in the count.

4.2 Implementation

In order to prepare the data for the experiment, the hashes from the largest partitions were computed and merged into a single file, which was then sorted in ascending order of hash value. The unique hashes from the file were then extracted; thus, all the `0x00`- and `0xFF`-filled sectors were automatically filtered out. The unique hashes were then sorted in order of ascending logical block addressing position and separated into individual files based on partition. The data in each partition was then divided into 128 equally-sized areas, each $\frac{1}{128}$ of the size of the partition. Following this, the probabilities of finding unique hashes in each area were computed as the number of unique hashes divided by the sizes of the areas in the sectors in each partition.

After the probability computations, the mean, median and standard deviation of the probabilities of unique hashes were calculated for each area of the partitions regardless of the partition sizes. The mean values were used to create a map of the storage media, showing where it is more likely to find user data (unique data) in a generic-NTFS formatted partition.

4.3 Evaluation

The map was evaluated by conducting an experiment that simulated a hash-based carving scenario and comparing the performance of sampling based on the map against that of a uniformly-distributed sampling. Four real partitions that were not included in the dataset were used as ground truth. The distributions of unique data in the four partitions were used to pick a random integer $target$. The map was then used to pick a random integer map and the uniform distribution was used to pick a random integer uni. All the random integers were selected in the same total range representing the logical block addressing positions of a fictitious partition, albeit with a bias for $target$ and map. The map received one hit when $map = target$ and the uniform distribution received one hit when $uni = target$. The predefined range was set to 16 MiB and divided into 128 equally-sized areas using the mapping process. The small partition size was chosen to increase the number of hits.

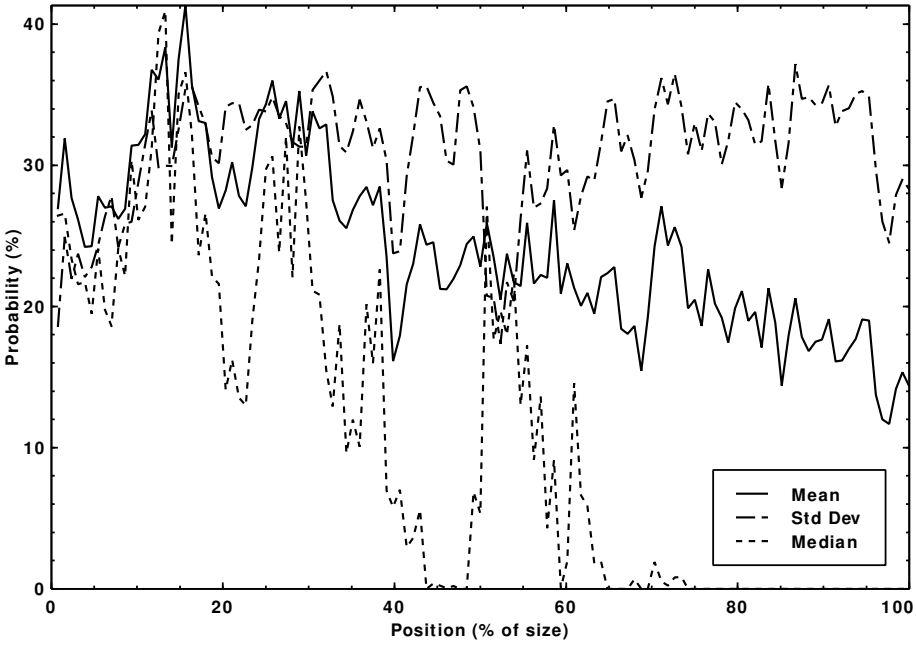

Figure 1. Probabilities of unique hashes.

The random sampling process was iterated 10^9 times for each of the four partitions to stabilize the results. The small number of partitions used to create the map adversely affected the evaluation results. Since the number of partitions used to create the ground truth was also small, the results were affected by individual variations in partition content. Another factor affecting the results is that the four partitions used as ground truth were taken from computers that were about to be scrapped; since their drives were well-used, the partitions had lower percentages of `0x00` and `0xFF` fillings towards their ends.

The experiment used Python 2.7 and the `random` library running on a Debian Stretch (version 9) computer.

5. Experimental Results

Figure 1 shows plots of the means, medians and standard deviations of the probabilities of unique hashes at different positions in the 30 partitions in the dataset. The positions are presented as percentages of the partition size. Each partition was split into 128 equally-sized areas based on the total size of the partition.

As seen in the map in Figure 1, the probabilities of unique hashes at different positions vary from approximately 12% to 41%. The low

median values in the second half of the partitions are due to the presence of `0x00` and `0xFF` fillings in a significant number of the partitions. Note that the median values are zero or close to zero because more than 50% of the partitions had no unique data or very small amounts of unique data.

An NTFS-formatted drive reserves 12.5% of its space for the MFT by default [49]. In all 30 partitions in the dataset, the MFT area started exactly 3 GiB into the partitions. Hence, the starting point of the area where non-resident file data is allocated is at the logical block addressing position:

$$P = 3 \cdot 2^{30} + 0.125 \cdot \textit{partition size}\,(\text{bytes})$$

However, this is not the case if the user changed the MFT reserved space at the time of formatting. In the case of a very small partition, the non-resident data allocation point would likely have been changed. Based on the dataset, the non-resident data allocation starting point is valid for partitions ≥ 60 GiB.

The bulk of the operating system, the first user files and software from the initial installation reside at the non-resident data allocation point. According to Microsoft [45–47], the minimum space requirement for Windows 7, 8, 8.1 or 10 installations is 20 GiB for 64-bit systems. Thus, the most probable starting point for day-to-day usage of a partition containing Windows is:

$$(3 + 20) \cdot 2^{30} + 0.125 \cdot \textit{partition size}\,(\text{bytes})$$

bytes into a partition. Converting the starting points to percentages of the associated partition lengths yields values between 14% and 43% (see Figure 1).

The largest number of operating system files were found in the beginning of the area and the number decreased towards the end. This explains the sharp negative trends in the plots between 20% and 40%, along with the peaks around 30%. In the case of the mean plot, from 40% onwards, the values slowly decrease and the standard deviations increase. This is due to differing usage patterns of the partitions – some partitions had been storing more data and/or had been more utilized than other partitions in the dataset.

A total of 3,809,786,792 unique hashes were found in the dataset; these correspond to data created locally by the user or operating system (e.g., logs). However, unique portions of MFT records were also in the data. Each file and directory is represented by at least one MFT record in NTFS. Depending on the number of file attributes, multiple MFT records may be needed to store the metadata. Typical examples are files with many alternate data streams or highly fragmented files.

```
Type: $DATA (128-12)   Name: N/A   Non-Resident  [...]
786432 786433 786434 786435 786436 786437 786438 786439
[...]
Type: $DATA (128-1)   Name: N/A   Non-Resident   [...]
786432 786433 786434 786435 786436 786437 786438 786439
[...]
Type: $DATA (128-6)   Name: N/A   Non-Resident   [...]
786432 786433 786434 786435 786436 786437 786438 786439
[...]
Type: $DATA (128-1)   Name: N/A   Non-Resident   [...]
786432 786433 786434 786435 786436 786437 786438 786439
[...]
Type: $DATA (128-6)   Name: N/A   Non-Resident   [...]
786432 786433 786434 786435 786436 786437 786438 786439
[...]
```

Figure 2. Portions of the $DATA attributes of `$MFT` files

.

MFT records may affect the results by increasing the number of unique non-user data hashes. To estimate the effect of the MFT records, the numbers of files and folders in 27 typical computers (office-grade and home-grade) were examined. The mean value was found to be 363,630 files, which corresponds to approximately 0.7% of the unique hashes in the 30 computers.

The `pagefile.sys` and `hiberfil.sys` files may also generate large numbers of unique hashes depending on how much they were used. These files certainly affect the map and the results. However, they should be included because they are of high value in a digital forensic investigation.

When studying the mapping process, four sectors were found to contain the same hash value at the same logical block addressing position in all the partitions in the dataset. The sectors, located in filesystem cluster 786,435, all contained the second half of MFT records that had been used only once according to their signature values [12]. The first part of the MFT records contained similar, but not equal information. The `istat` tool [13] revealed that the sectors belonged to the `$MFT` file (i.e., filesystem itself). The $DATA attributes of the `$MFT` files in the five computers to which raw access was still available allocated the same eight clusters at the beginning of the run length. Note that the eight numbers in every third row in Figure 2 indicate the filesystem clusters allocated to a file. Filesystem cluster 786,435 contained four static sectors – at positions 6,291,481, 6,291,483, 6,291,485 and 6,291 487 – that were found in all 30 partitions.

Table 2. Evaluation results.

Hits (Map)	Hits (Uniform)	Percentage (Map/Uniform)
28,635	30,279	94.6%
29,881	30,363	98.4%
32,556	30,836	105.6%
33,257	30,461	109.2%
124,329	121,939	102.0%

Upon combining this with the static content of the four sectors in cluster 786,435, the NTFS formatting appears to place the start of the MFT at the same position – exactly 3 GiB into the partition. If this is true, then the first and last sectors of an NTFS partition should contain the hex string `00 00 0C 00 00 00 00 00` starting at position `0x30` (little endian) [44]. This was verified for the five computers for which raw access was still available.

According to Carrier [12], the $DATA attribute of the `$MFTMirr` file allocates clusters in the middle of a filesystem. This implies that the middle sector, based on the size of the volume (partition), is actually where the mirror should be kept. However, this was not always true. In four of the five computers to which full access was available, the `$MFTMirr` file allocated filesystem cluster 2 and, in the last partition, filesystem cluster 8,912,895 was allocated. However, the latter partition was 59,919,808 clusters in size; hence, none of the `$MFTMirr` files were located near the middle of any of the partitions. Consequently the NTFS allocation strategy appears to have changed since Carrier published his book [12].

In order to evaluate the efficiency of the map in random sampling situations, four NTFS partitions (different from the 30 partitions in the mapping dataset) were used to create the map. The four 16 MiB partitions were divided into 128 equally-sized areas that were sampled 10^9 times in the evaluation.

Table 2 shows the evaluation results. In particular, the table shows the number of hits using the map relative to using a uniformly-distributed sampling rate. Due to the small number of partitions used in the evaluation, the distributions of unique data in the individual partitions have a high impact on the result. Therefore, the evaluation results are merely the first indicator of the performance of future maps, not the final answer (future research will conduct a new evaluation using a larger number of

partitions). In any case, the best result – when the map most resembles one of the evaluation partitions – is almost 10% better than using a uniformly-distributed sampling rate. Changing the number of equally-sized areas does not change the results in any significant way. Changing the 16 MiB partition size also does not change the results.

6. Discussion

The concept of searching for data where it is more likely to be found is more appealing than randomly searching for data. Nevertheless, an empirical evaluation was conducted to test the implementation. The evaluation reveals a 2% improvement when using the map compared with using a uniformly-distributed sampling rate. This is certainly not a paradigm shift, but it is a positive indicator of the utility of the concept.

The reason for the modest result is the small number of partitions used to create the map. More partitions are required to reveal the underlying deterministic allocation pattern; this would also provide a solid statistical foundation and improve the strength of the results. Having a large enough dataset would enable its division into several use cases, each producing its own map. An example would be to differentiate between web surfers, office administrators and file sharers. However, such a study would require a much larger data collection effort while maintaining a high level of control of the collected material to filter unique data that was not created by the user or system (e.g., data written during disk wiping).

After the mapping foundation is stable, the efficiency of digital forensic methods and tools – especially in file carving situations where a filesystem is not present – can be improved in several ways. One example is when using hash-based carving to find portions of files on storage media. In this case, the following three strategies are possible:

- **Speed is Prioritized:** The total number of samples is reduced compared with the uniformly-distributed sampling case without any significant loss in detection rate. This strategy could be used in triage situations or when preliminary results are required quickly.
- **Speed is Maintained:** The same amount of samples are maintained compared with the uniformly-distributed sampling case, which enhances detection at the same execution speed. This standard case can be used without changing the digital forensic process.
- **Detection Rate is Prioritized:** A larger number of samples are used than in the case of uniformly-distributed sampling, yielding a much higher detection rate and a lower cost in terms of execu-

tion speed. An example is a situation where a suspect's drive has an unusual usage pattern. The standard number of hashes can be maintained in low priority areas of a drive whereas a higher sampling rate can be used in high priority areas for improved detection.

When the area reserved for the MFT is used up, a new area amounting to 12.5% of the volume size is added. This area may be contiguous, but it does not have to be so. As the filesystem grows, new MFT records are added and allocated where suitable [12]. Thus, an old and well-used NTFS partition may very well have MFT records spread all over the storage space. This would possibly affect the creation of the map, adding noise to the unique data.

According to the empirical study of the number of files and directories (usually represented by a single MFT record each) in an NTFS partition, the amount of MFT records corresponds to approximately 0.7% of the total amount of unique hashes in each partition. The actual amount of unique hashes belonging to an MFT record is probably less than 0.7% because the second part of an MFT record often contains 510 zero-bytes followed by a two-byte signature value at the end of the sector. Signature values are used by NTFS to verify the integrity of data structures (but not sectors containing file content) that span two or more sectors [12]. The last two bytes of every sector in such a data structure are called a fixup value and are moved to an array in the beginning of the structure during the process of writing to disk. The last two bytes are then replaced by the signature value. When the data structure is read, the signature values are used to check that all the sectors that are read have the same signature value and, thus, belong to the same data structure. The signature value is incremented by one every time a data structure is updated on disk [12].

The worst case scenario is a partition filled with files that are less than approximately 700 B in size, resulting in a partition filled with MFT records that store the data internally. The maximum size of an internal $Data attribute varies depending on the sizes of other attributes stored in the MFT record. Most sources give a maximum internal $Data attribute size of 600 to 700 bytes; Microsoft reports a 900 B limit [49]. If all the files contain the same data, only the MFT metadata (timestamps, etc.) would differ; thus, the partition would still appear to be filled with random data. The maximum number of files in an NTFS partition is $2^{32} - 1$ [49], so the partition would be approximately 4 TiB in size.

Another way to estimate the number of unique MFT record hashes in the dataset is to generate SHA-1 hash values for all possible combinations of 510 zeros and a two-byte signature value, which correspond to the

second half of a standard MFT record. The first such hash that is unique in the dataset represents a signature value of 3613 (`0x1D0E` little endian). Many of the lower signature values generate several thousand hits. There is no guarantee that all the generated hashes belong to MFT records; however, at least four do and, consequently, the percentage of unique MFT hash values that pollute the dataset is likely less than 0.7%. Hence, the unique hashes of the MFT records do not significantly affect the precision of the map.

The experiment was limited to computers running Microsoft Windows 7 and later versions with NTFS-formatted main partitions. The privacy of the computer owners was protected by using SHA-1 hashes to obscure the real data. This limited the ability to trace the original data corresponding to each hash. However, since the focus is on the logical block addressing positions of unique hashes, the precise data represented by the hashes is not needed for map creation.

The proposed method can also be used to find shared data. Of special interest is static data – shared data found in the same logical block addressing position in unrelated storage media. Knowledge of the logical block addressing positions of static data is of value in several digital forensic applications. For example, prioritizing search in forensic imaging and analysis could provide an analyst with the means to break drive encryption via a plaintext attack [60] (depending, of course, on the encryption algorithm that is used).

The logical block addressing position of static data can be used to handle corrupt storage media. In many cases, large portions of corrupt media are readable, but there are no indications of the forensic value of the lost portions. Having access to a map of static content in storage media can help a digital forensic analyst improve the evaluative reporting in casework by indicating the forensic value of lost areas. This contributes to higher confidence in the collected evidence.

Furthermore, a map can be used to create signatures that identify the filesystems in partially-recovered partitions. These signatures are feasible because the metadata layout and allocation process during installation differ for operating systems.

Finally, media areas that should have high probabilities of static content but do not have this content could be indicators of the presence of malware or other suspicious activities because deviations are unlikely in such areas. Instead of having to hash every file in a filesystem in search of deviations, the search can start at the most likely place in the filesystem. The partition is then scanned in descending order of probability of static content.

7. Conclusions

The proposed method for hash-based file carving is based on the principle that it is better to search for data in locations where the probability of finding the data is high. The method relies on a map of the probability of finding unique data at various logical block addressing positions in storage media. Unique data is data that is created locally on a computer and not (yet) shared. This includes system-created data such as log files and user-created local data that is not downloaded from the Internet (downloaded data corresponds to shared data). Uniquely-created data is often more valuable in a forensic investigation than shared data, although shared data (e.g., child abuse material) can be valuable too.

The map provides a digital forensic analyst with a pre-computed view of storage media that enables the forensic process to focus on the most relevant portions of media instead of spending valuable time scanning the entire storage media from beginning to end. Unique data is only used to create the map; this is done once, although regular updates to the map are recommended. After the map is created, it can be used repeatedly for any data or with any method, tool or investigative process without having to recreate the map for every new case.

Creating a probability map of unique (or static) data at different positions in storage media opens up a host of applications. The map could be used when performing triage, planning the order of analysis of large amounts of seized storage media, estimating the value of partially-analyzed data due to corruption and breaking the encryption of storage media.

Future research will extend the dataset to stabilize map creation and make the map more reliable. It will also explore other methods for creating maps as well as creating separate maps for use cases. Research will also search for and study the origin of interesting areas in storage media such as the four sectors with the same hash values found at approximately 3 GiB into all 30 partitions, which indicates that certain areas of NTFS partitions are static. Finally, future work will extend the method to other filesystems, especially Ext4 and Apple File System (APFS), with the goal of creating a general mapping process for storage media, regardless of type and filesystem.

The authors of this chapter are interested in releasing the current hash dataset to the public. However, due to its size, the optimal transfer option will have to be discussed with the interested party. Interested parties are encouraged to contact the first author (`martin@filecarving.net`) to arrange for file transfers.

Acknowledgement

This research was sponsored by the Norwegian Research Council Ars Forensica Project No. 248094/O70.

References

[1] I. Ahmed, K. Lhee, H. Shin and M. Hong, On improving the accuracy and performance of content-based file type identification, *Proceedings of the Fourteenth Australasian Conference on Information Security and Privacy*, pp. 44–59, 2009.

[2] S. Axelsson, The normalized compression distance as a file fragment classifier, *Digital Investigation*, vol. 7(S), pp. S24–S31, 2010.

[3] S. Axelsson, Using normalized compression distance for classifying file fragments, *Proceedings of the International Conference on Availability, Reliability and Security*, pp. 641–646, 2010.

[4] S. Axelsson, K. Bajwa and M. Srikanth, File fragment analysis using normalized compression distance, in *Advances in Digital Forensics IX*, G. Peterson and S. Shenoi (Eds.), Springer, Berlin Heidelberg, Germany, pp. 171–182, 2013.

[5] J. Barbara, Solid state drives: Part 5, *Forensic Magazine*, vol. 11(1), pp. 30–31, 2014.

[6] F. Breitinger and K. Petrov, Reducing the time required for hashing operations, in *Advances in Digital Forensics IX*, G. Peterson and S. Shenoi (Eds.), Springer, Berlin Heidelberg, Germany, pp. 101–117, 2013.

[7] F. Breitinger, C. Rathgeb and H. Baier, An efficient similarity digests database lookup – A logarithmic divide and conquer approach, *Journal of Digital Forensics, Security and Law*, vol. 9(2), pp. 155–166, 2014.

[8] F. Breitinger, G. Stivaktakis and H. Baier, FRASH: A framework to test algorithms of similarity hashing, *Digital Investigation*, vol. 10(S), pp. S50–S58, 2013.

[9] C. Buckel, Understanding Flash: Blocks, Pages and Program Erases, *flashdba Blog* (`flashdba.com/2014/06/20/understanding-flash-blocks-pages-and-program-erases`), June 20, 2014.

[10] C. Buckel, Understanding Flash: The Flash Translation Layer, *flashdba Blog* (`flashdba.com/2014/09/17/understanding-flash-the-flash-translation-layer`), September 17, 2014.

[11] W. Calhoun and D. Coles, Predicting the types of file fragments, *Digital Investigation*, vol. 5(S), pp. S14–S20, 2008.

[12] B. Carrier, *File System Forensic Analysis*, Pearson Education, Upper Saddle River, New Jersey, 2005.

[13] B. Carrier, TSK Tool Overview (`wiki.sleuthkit.org/index.php?title=TSK_Tool_Overview`), January 13, 2014.

[14] T. Chung, D. Park, S. Park, D. Lee, S. Lee and H. Song, A survey of the flash translation layer, *Journal of Systems Architecture*, vol. 55(5-6), pp. 332–343, 2009.

[15] S. Collange, Y. Dandass, M. Daumas and D. Defour, Using graphics processors for parallelizing hash-based data carving, *Proceedings of the Forty-Second Hawaii International Conference on System Sciences*, 2009.

[16] Cryptology Group at Centrum Wiskunde and Informatica and Security, Privacy and Anti-Abuse Group at Google Research, SHAttered – We have Broken SHA-1 in Practice (`shattered.io`), 2017.

[17] Y. Dandass, N. Necaise and S. Thomas, An empirical analysis of disk sector hashes for data carving, *Journal of Digital Forensic Practice*, vol. 2(2), pp. 95–104, 2008.

[18] Digital Corpora, Real Data Corpus (`digitalcorpora.org/corpora/disk-images/real-data-corpus`), July 15, 2018.

[19] EUROPOL: European Law Enforcement Agency, IOCTA 2016: Internet Organized Crime Threat Assessment, Technical Report, European Police Office, The Hague, The Netherlands, 2016.

[20] K. Fairbanks, An analysis of Ext4 for digital forensics, *Digital Investigation*, vol. 9(S), pp. S118–S130, 2012.

[21] K. Fairbanks, A technique for measuring data persistence using the Ext4 file system journal, *Proceedings of the Thirty-Ninth Annual IEEE Computer Software and Applications Conference*, vol. 3, pp. 18–23, 2015.

[22] K. Fairbanks and S. Garfinkel, Column: Factors affecting data decay, *Journal of Digital Forensics, Security and Law*, vol. 7(2), pp. 7–10, 2012.

[23] S. Fitzgerald, G. Mathews, C. Morris and O. Zhulyn, Using NLP techniques for file fragment classification, *Digital Investigation*, vol. 9(S), pp. S44–S49, 2012.

[24] K. Foster, Using Distinct Sectors in Media Sampling and Full Media Analysis to Detect Presence of Documents from a Corpus, Master's Thesis, Department of Computer Science, Naval Postgraduate School, Monterey, California, 2012.

[25] 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.

[26] S. Garfinkel, A. Nelson, D. White and V. 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.

[27] S. Gibbs, From Windows 1 to Windows 10: 29 years of Windows evolution, *The Guardian*, October 2, 2014.

[28] P. Gladyshev and J. James, Decision-theoretic file carving, *Digital Investigation*, vol. 22, pp. 46–61, 2017.

[29] Y. Gubanov and O. Afonin, Why SSD drives destroy court evidence and what can be done about it, *Forensic Focus*, October 23, 2012.

[30] Y. Gubanov and O. Afonin, Recovering evidence from SSD drives in 2014: Understanding trim, garbage collection and exclusions, *Forensic Focus*, September 23, 2014.

[31] Y. Gubanov and O. Afonin, SSD and eMMC forensics 2016, *Forensic Focus*, April 20, 2016.

[32] Y. Gubanov and O. Afonin, SSD and eMMC forensics 2016 – Part 2, *Forensic Focus*, May 4, 2016.

[33] Y. Gubanov and O. Afonin, SSD and eMMC forensics 2016 – Part 3, *Forensic Focus*, June 7, 2016.

[34] Guidance Software, File Block Hash Map Analysis, Version 8.8.5, Waterloo, Canada (`www.guidancesoftware.com/app/File-Block-Hash-Map-Analysis`), 2018.

[35] J. Jones, T. Khan, K. Laskey, A. Nelson, M. Laamanen and D. White, Inferring previously uninstalled applications from residual partial artifacts, *Proceedings of the Eleventh Annual Conference on Digital Forensics, Security and Law*, pp. 113–130, 2016.

[36] M. Karresand, Completing the Picture – Fragments and Back Again, Licentiate Thesis, Institute of Technology: Faculty of Science and Engineering, Linkoping University, Linkoping, Sweden, 2008.

[37] M. Karresand and N. Shahmehri, File type identification of data fragments by their binary structure, *Proceedings of the Seventh Annual IEEE SMC Information Assurance Workshop*, pp. 140–147, 2006.

[38] M. Karresand and N. Shahmehri, Oscar – File type and camera identification using the structure of binary data fragments, *Proceedings of the First Conference on Advances in Computer Security and Forensics*, pp. 11–20, 2006.

[39] M. Karresand and N. Shahmehri, Oscar – File type identification of binary data in disk clusters and RAM pages, *Proceedings of the Thirty-First IFIP TC-11 International Information Security Conference*, pp. 413–424, 2006.

[40] M. Karresand and N. Shahmehri, Oscar – Using byte pairs to find the file type and camera make of data fragments, *Proceedings of the Second European Conference on Computer Network Defense*, pp. 85–94, 2007.

[41] M. Karresand and N. Shahmehri, Reassembly of fragmented JPEG images containing restart markers, *Proceedings of the Fourth European Conference on Computer Network Defense*, pp. 25–32, 2008.

[42] J. Kornblum, Identifying almost identical files using context triggered piecewise hashing, *Digital Investigation*, vol. 3(S), pp. S91–S97, 2006.

[43] Q. Li, A. Ong, P. Suganthan and V. Thing, A novel support vector machine approach to high entropy data fragment classification, *Proceedings of the South African Information Security Multi-Conference*, pp. 236–247, 2010.

[44] LSoft Technologies, NTFS Partition Boot Sector, Mississauga, Canada (`www.ntfs.com/ntfs-partition-boot-sector.htm`), 2018.

[45] Microsoft, Windows 7 System Requirements, Redmond, Washington (`support.microsoft.com/en-us/help/10737/windows-7-system-requirements`), April 12, 2017.

[46] Microsoft, Windows 8.1 System Requirements, Redmond, Washington (`support.microsoft.com/en-gb/help/12660/windows-8-system-requirements`), April 12, 2017.

[47] Microsoft, Windows 10 System Requirements, Redmond, Washington (`support.microsoft.com/en-us/help/4028142/windows-windows-10-system-requirements`), November 20, 2017.

[48] Microsoft, Default Cluster Size for NTFS, FAT and exFAT, Redmond, Washington (`support.microsoft.com/en-us/help/140365/default-cluster-size-for-ntfs--fat--and-exfat`), April 17, 2018.

[49] Microsoft, How NTFS Works, Redmond, Washington (`technet.microsoft.com/pt-pt/library/cc781134(v=ws.10).aspx`), October 28, 2018.

[50] Net Applications, Desktop Operating System Market Share, Aliso Viejo, California (`www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0`), 2017.

[51] A. Pal and N. Memon, The evolution of file carving, *IEEE Signal Processing*, vol. 26(2), pp. 59–71, 2009.

[52] R. Poisel, M. Rybnicek and S. Tjoa, Taxonomy of data fragment classification techniques, in *Digital Forensics and Cyber Crime*, P. Gladyshev, A. Marrington and I. Baggili (Eds.), Springer, Cham, Switzerland, pp. 67–85, 2014.

[53] R. Poisel and S. Tjoa, A comprehensive literature review of file carving, *Proceedings of the International Conference on Availability, Reliability and Security*, pp. 475–484, 2013.

[54] D. Quick and K. Choo, Data reduction and data mining framework for digital forensic evidence: Storage, intelligence, review and archive, *Trends and Issues in Crime and Criminal Justice*, no. 480, pp. 1–11, September 2014.

[55] D. Quick and K. Choo, Impacts of increasing volume of digital forensic data: A survey and future research challenges, *Digital Investigation*, vol. 11(4), pp. 273–294, 2014.

[56] D. Quick and K. Choo, Big forensic data reduction: Digital forensic images and electronic evidence, *Cluster Computing*, vol. 19(2), pp. 723–740, 2016.

[57] R. Reiter, T. Swatosh, P. Hempstead and M. Hicken, Accessing logical-to-physical address translation data for solid state disks, U.S. Patent No. 8898371, November 25, 2014.

[58] V. Roussev, Managing terabyte-scale investigations with similarity digests, in *Advances in Digital Forensics VIII*, G. Peterson and S. Shenoi (Eds.), Springer, Berlin Heidelberg, Germany, pp. 19–34, 2012.

[59] N. Rowe, Identifying forensically uninteresting files using a large corpus, in *Digital Forensics and Cyber Crime*, P. Gladyshev, A. Marrington and I. Baggili (Eds.), Springer, Cham, Switzerland, pp. 86–101, 2014.

[60] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*, John Wiley and Sons, Hoboken, New Jersey, 1996.

[61] M. Stevens, E. Bursztein, P. Karpman, A. Albertini and Y. Markov, The first collision for full SHA-1, *Proceedings of the Thirty-Seventh Annual International Cryptology Conference*, pp. 570–596, 2017.

[62] A. Tridgell, `spamsum` (`www.samba.org/ftp/unpacked/junkcode/spamsum/README`), July 27, 2015.

[63] R. van Baar, H. van Beek and E. van Eijk, Digital forensics as a service: A game changer, *Digital Investigation*, vol. 11(S1), pp. S54–S62, 2014.

[64] H. van Beek, E. van Eijk, R. van Baar, M. Ugen, J. Bodde and A. Siemelink, Digital forensics as a service: Game on, *Digital Investigation*, vol. 15, pp. 20–38, 2015.

[65] C. Veenman, Statistical disk cluster classification for file carving, *Proceedings of the Third International Symposium on Information Assurance and Security*, pp. 393–398, 2007.

[66] 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.