Chapter 1

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

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

Abstract Digital forensics, and especially file carving, is burdened by the large and increasing amount of data that needs to be processed. Attempts to solve the problem are being made by introducing for example more efficient carving algorithms, parallel processing in the cloud, and the reduction of data by filtering out uninteresting files.

We propose to use the principle of searching where it is more probable to find what you are looking for. This is done by creating a map of the probability of finding unique data at different Logical Block Addressing (LBA) positions of a collection of storage media. We use Secure Hash Algorithm 1 (SHA-1) hashes of 512 B sectors to represent the data. Our results show that the mean probability of finding unique hash values at different LBA positions vary between 12% to 41% over a New Technology File System (NTFS) partition.

The map can be used by a forensic investigator to prioritize relevant areas in storage media, without the need for a working file system. It can also be used to increase the efficiency of hash-based carving by dynamically changing the random sampling frequency, which we show. Our method also contributes to the digital forensics processes in general, which can now be focused on the interesting regions on storage devices, increasing the probability of getting relevant results faster.

Our work is based on a collection of 30 NTFS partitions from computers running Microsoft Windows 7 and newer.

Keywords: Digital forensics, file carving, partition content map, hash-based carving, NTFS

1. Introduction

The ever-increasing amount of data to be handled in digital forensics is a major challenge to digital forensics case work [55] and has been

discussed for many years [28][19][54][6][58]. The field of *file carving* is especially affected by the increasing amount of data. File carving is used in situations where there is no file system available, instead only the the properties of the stored data itself [52][51] are used. That principle connects this article to our previous work on determining the data type (file type) of fragmented data by using histograms of the frequency of bytes, byte pairs and the difference between consecutive byte values [35] [40][36][39][37][38]. We have also used the compressibility of data for type identification [3][2][4]. As before we now use small blocks of data (512 B sectors in this article) and their statistical properties to improve file carving, but now we apply the principle of finding patterns in unknown data to full hard disk partitions. However, this time we determine the most probable position of user data, not the exact type of it.

Being able to carve files without the help of a working file system is difficult, but highly valuable to the digital forensic investigator. The research community is therefore focused on solving the problem of the increasing amount of data by different means. In a survey from 2014 Quick and Choo [55] lists the following concepts; data mining, data reduction and subsets, triage, intelligence analysis and digital intelligence, distributed and parallel processing, visualization, digital forensics as a service (DFaaS) and different artificial intelligence techniques.

In the file carving sub-field of *hash-based carving*, hashes of blocks of unknown data from the storage media is compared to known equally sized blocks of suspicious material. The large amount of comparisons of hashes made by a hash-based carving algorithm puts extra burden on the forensic process. Therefore different strategies, techniques and algorithms for hash-based carving have been developed [26][8][7][66][24] [25][15].

What has not yet been tested is to utilize the principle of searching for something where it is more probable to find it. Since the allocation algorithm of the operating system (OS) will place new data in the file system according to a set of rules, not randomly, the principle can be used in the digital forensic field too. The allocation process is however too complex to fully understand and thus commonly regarded as random. Therefore the current principle is to linearly search the storage media from beginning to end, regardless of the most probable position of the sought-after data.

Most of the data of interest in an investigation is related to user activity, i. e. system logs and files created by the user. Such data is often unique to the specific computer and cannot be found elsewhere, because the probability of two users independently creating exactly the same data is negligible. Of course also shared data downloaded (from

the internet or elsewhere) by the user are of interest, for example child abuse material. Such data will be stored intertwined with the unique user data according to the allocation algorithm's rules. Hence it makes sense to use the Logical Block Addressing (LBA) position of unique user data to also find where the user's activity has stored shared data.

We have therefore performed an experiment using Secure Hash Algorithm 1 (SHA-1) hashes of the content of non-related computers running Windows 7 and later, to find the probability of unique hashes (unique user data) at different positions in the 30 largest New Technology File System (NTFS) formatted partitions of 26 hard disks. The data was chosen to be as realistic as possible to increase the applicability of our results, and we therefore used real world computers in the data collection. The unique data are unique in our data set, there is no guarantee that they are unique world wide.

The rest of this paper is organized as follows: The remaining parts of Section 1.1 presents related work and our contributions. In Section 1.2 we describe our data set and how it was collected, together with a description of how the experiments were implemented. Section 1.3 presents the results of the study. In Section 1.4 we discuss the effects and implications of our results to the research field of hash-based carving and also to other areas within and related to digital forensics. Section 1.5 concludes the work and presents ideas of future work to be done.

1.1 Related work

Although we have not found any work that directly relates to our work, there are several research sub-fields that have bearing on our work; The main field being file carving, and especially its sub-field of hash-based carving.

1.1.1 File fragment carving. Apart from our work within the file fragment carving area there are also work done by others using different means to identify the type of data fragments. Veenman [65] use the entropy, histogram and Kolmogorov complexity of 4 KiB file fragments to determine their type. The result show that histograms have the highest detection rate versus false positives of the chosen algorithms. Calhoun and Coles [11] experiment with different statistical measures, for example frequency of ASCII codes, entropy, modes, mean, standard deviation and correlation between adjacent bytes. They also look at using the longest common sub-strings and sub-sequences between file fragments for data classification. Ahmed et al. [1] use the byte frequency distribution with a new method of measuring the distance be-

tween the statistical properties of a data fragment and a model. Instead of using the Mahalanobis distance measure they use cosine similarity with improved results. Li et al. [43] also use the byte frequency distribution (histogram) of different data fragments, but in conjunction with a support vector machine as a discriminant between different data types. They explain that the best results are achieved using the byte frequency distribution alone. Fitzgerald et al. [23] combine several statistical measures of data fragments (among them histograms of one and two byte sequences, entropy and Kolmogorov complexity) to get feature vectors that are fed into a support vector machine for classification. They notice that their method outperform many method presented in previous work. However, they do not evaluate the contribution of each of the chosen feature vectors, but instead leave it as future work. There is also a taxonomy of data fragment classification techniques by Poisel et al. [53] describing the research area.

1.1.2 Hash-based carving. The digital forensics research field of hash-based carving compares hashes of known file blocks to hashes of equally sized blocks from a suspects hard drive. In that way even files that are partially overwritten or damaged can be identified.

The roots of the research field can be traced back to the `spamsum` tool by Tridgell [62]. According to Garfinkel one of the first times hashes are used for file carving is during the Digital Forensic Research Workshop (DFRWS) 2006 Carving Challenge [26]. Later the `spamsum` tool is used as a basis for an article by Kornblum [42] on piecewise hashing and what is now known as *approximate matching*. The concept of using hashes for file carving is further studied by Dandass [18] in 2008 in an article presenting an empirical analysis of disk sector hashes. The term hash-based carving is first introduced by Collange et al. [15] exploring the possibility of using a Graphics Processing Unit (GPU) for comparing hashes of 512 byte sections of known files with hashes of 512 byte sectors from disk images.

When Garfinkel use hashes for file carving in the DFRWS 2006 Carving Challenge [26] parts of files found on the internet are hashed and used to find equal hashes in the challenge image. These experiences lead to the development of the `frag_find` tool [25]. In connection to the `frag_find` article the authors discuss the optimal size of the data blocks to hash. They conclude that the size shall be equal to the sector size, without stating if they mean 512 B or 4 KiB sectors. Garfinkel et al. [26] elaborate further on the size of hashed blocks and state that starting with Windows NT 4.0 the default minimum allocation unit in NTFS is 4 KiB [44].

Foster [24] discusses the problem of data shared across files, stating that "the block of NULs is the most common block in our corpus" [24, p. 15], relating them to the NULL padding of files. The problem of the large amount of data to handle is also discussed. Young et al. [66] continues the work further developing the Foster's ideas. The authors discuss the optimal block size, how to handle a large amount of data, efficient hash algorithms, good data sets to use and common blocks of files.

Random sampling is used to improve the speed of hash-based carving in several articles [26][24][25]. To find a suitable sampling frequency the problem is regarded as sampling without replacement. Using a higher sampling frequency may increase the detection rate, but has a negative impact on the execution speed. The problem is to find a suitable balance between the two alternatives.

1.1.3 Data persistence. The concept of data persistence is interesting to our work because the persistence at different areas of storage media indicates that they are not reused. This information is valuable when creating a map of a generic storage media.

Jones and Khan [34] have created a framework to enable studies of (deleted) file persistence in storage media. They use differential forensic analysis to compare snapshots of file systems in use and follow the decay of deleted files over time.

Fairbanks and Garfinkel [20] present 12 factors affecting data persistence in storage media. Fairbanks [21][22] also describes the low-level functions of ext4 and their effect on digital forensics.

1.1.4 Data reduction. Quick and Choo propose different methods to reduce the amount of data needed to be analyzed in digital forensic investigations. Their approach [56][54] builds on extracting specific files using a list of key files and then work on the subset of files. This requires a working file system, limiting the methods applicability. Also the list of key files needs to be constantly updated.

Rowe [59] has a similar approach as Quick and Choo, although more technical. He compares nine methods for identifying uninteresting files, defined as "those files whose contents do not provide forensically useful information about users of a drive." [59, p. 86]. However, the methods studied by Rowe all require a working file system, which is not consistent with the foundation of file carving.

1.1.5 Data mapping. Key [41] has developed an EnScript module to the EnCase software which creates a map of the recoverable

sectors of a file found in a file system. It can handle situations where other tools does not work, for example partially damaged files, although it is very processor intensive and therefore can only create maps of a few files at a time.

Gladyshev and James [28] study the problem of file carving from a decision-theoretic point of view. They suggest a model where storage media is sampled with a frequency based on different properties of the hard disk and the file type that is to be found. In some specific situations their carving model outperforms standard linear carving algorithms, but their solution is not yet generally applicable. Gladyshev and James mention using the distribution of data on disk, but do not seem to relate that to the probability of finding user data at different LBA positions in storage media.

In two articles by van Baar et al. [63] and van Beek et al. [64] outlining the DFaaS system Hansken [64] and its predecessor Xiraf [63] the concept of non-linear extraction of data from images is discussed. Both van Baar and van Beek suggests that the Master File Table (MFT) records (the file system meta data) of an NTFS partition are extracted first. The MFT records are then used to find other interesting areas in the file system. van Baar and van Beek also suggest that the analysis process is used to influence the imaging process by having specified parts being prioritized.

1.2 Contribution

As can be seen from the review of related work, there is a need to improve the efficiency of the tools and algorithms used in digital forensics, and especially in file carving. There are many different proposed solutions to the problem, but no one has yet utilized the inherent structures of the allocation algorithms to address the problem. We therefore present the novel idea of using the probability of finding user data at different locations in storage media to govern the digital forensic process and hence enabling an immediate increase of the efficiency of existing file carving algorithms and tools. In hash-based carving the concept can be used to increase the efficiency when doing random sampling by varying the sampling rate in accordance with the probability of finding user data at different LBA positions. The principle can also be used during triage and other situations where speed and detection rate has to be balanced.

Unlike many of the methods presented in related work our method works without a file system. The map we create can be used directly to further improve the speed of any of the file carving algorithms presented as related work by showing the most probable position of unique data

in a general NTFS formatted storage media. It can either be used for starting the forensic process at the position with the highest probability of containing data of interest (for example user data) or varying the sampling rate in accordance with the probability of finding user data. In the latter case the sampling frequency will be higher where it matters most and lower in other areas, increasing the probability of getting a hit while maintaining the same amount of samples as with equally distributed sampling.

Our work also benefits the digital forensic investigator, because our map introduces the possibility to plan the forensic process in a similar way to how a map is used when planning operations in the physical world. Currently storage media are treated as black boxes, forcing the forensic investigators to spend valuable time scanning them from start to end before the analysis. This is especially useful in general file carving situations when there is no file system to govern the search. With our method the forensic investigators can focus on relevant areas of the storage media and postpone, or even skip, less relevant areas.

The map can also be used in storage media imaging situations. By starting the imaging process at the most probable position of user data, continuing in decreasing order of relevance, the analysis process can be run almost in parallel to the imaging since the most relevant data for analysis will be immediately available. In that way the analysis process can be started earlier, even before the imaging is finalized, saving valuable time and effort for the forensic investigators. Of course the reliability of the analysis will increase as more data are analyzed, but a preliminary result to guide the progressing work will be available at an earlier stage. This concept is also supported by the Hansken project [64] [63]. By implementing our concept in Hansken its ability to also handle media with broken file systems will be higher, possibly close to the performance of the standard process.

To enable handling of any storage media, regardless of its file system cluster size, our method use 512 byte sectors when hashing the data. Since our map is created once and can be reused there is no performance penalty in using it, just like a physical map. Since we have divided the map into a small number of equally sized areas (currently 128) any random seek penalty will only occur between these areas, not within, and thus can be ignored. Also the only situation where the use of 512 byte hashes are required is when the map is created. There is no need to use 512 byte hashes when performing case work on a suspect's hard drive. Likewise any hashing algorithm can be used for case work because the hashes of the map are only used to calculate the probability of user data at different positions and never meant to be compared to hashes from a

specific case. If a hash algorithm is broken it can simply be exchanged for a new and better algorithm, our map will still work.

During our work we found a total of nine sectors having the same hash value at the same LBA position in all 30 partitions. These sectors can for example be used to identify an NTFS file system, find the start of a NTFS partition and locate the $MFT file for further processing. This can be done regardless of the state of the file system.

To the best of our knowledge this specific research field has not yet been explored, a field with the possibility to bring improvements to a number of related research fields in digital forensics. This new approach therefore has a high impact factor and relevance to most, if not all, digital forensic cases.

2. Experimental Setup

To determine the distribution of unique data in the major NTFS formatted partition of a common Microsoft Windows computer we first collect live data from real computers. Then we calculate the probability of finding unique hash values at different LBA positions. Finally we create a map by calculating the mean probability of a number of (128 in our case) equally sized partition areas based on LBA position. The mean probability calculation is done to generalize and scale the map into a usable format.

To lower the size of data to be stored for the experiment and also to protect the privacy of the user we use the SHA-1 algorithm to hash each 512 byte sector of all 30 NTFS formatted main partitions included in the experiment. We use SHA-1 because it currently offers the best balance between speed, collision risk and hash size among the hash algorithms we choose from (Message-Digest algorithm 5 (MD5) and the SHA family). The choice is based on a practical evaluation using available hardware. The hashing of data is done locally at each source computer and thus only the resulting hashes leave the computers.

SHA-1 maps 512 bytes of data onto a 20 byte long hash and thus there is a theoretical risk of collisions. If we apply the Birthday Paradox to our situation, the risk of a collision is approximately $1.1 \cdot 10^{-28}$ and hence negligible[1]. We therefore assume a unique SHA-1 hash to represent a unique piece of data.

Even though the SHA-1 algorithm is broken [61][17] from a cryptographic point of view the risk of an intentional collision is also negligible, because the amount of computing power required to create a collision is out of reach for the common user [61][17]. Also such an attack would require an attacker to create a large amount of collisions for a majority

of the storage media in the map source data. It would be much simpler to fill the disks with shared and unique data in an intentional pattern. This is however mitigated by collecting the source data from non-related sources. Finally the mapping process is not limited to the use of SHA-1, any hashing algorithm will do, as long as all mapping data is hashed using the same algorithm.

2.1 Data Collection

To get hold of data representing real life situations we chose to use a convenience sample collecting data from computers owned by people in our acquaintanceship. We did not use the Real Data Corpus (RDC) because the time stamps on the RDC web site [16] indicate that the last update of the data set was made in 2011. Therefore our data set is more up to date containing also versions 8 and 10 of Windows[2].

We have collected data from 30 partitions of 26 computers (23 consumer grade and 3 office grade). The data was collected by hashing every 512 byte sector of the entire hard disks using the `dcfldd` disk imaging tool set to use the SHA-1 cryptographic hash algorithm. The OS installations represent three different language packs and range from Microsoft Windows 7 to Windows 10, both Enterprise, Professional, Ultimate, Home and Educational versions. Some of the computers have been upgraded form an earlier Windows version to Windows 10. Five of the computers are in our possession and we therefore have access to their raw content.

The reason for using real computers and not a simulation in a laboratory environment is to avoid any bias from the simulation of user behavior. By using real computers our results will be as close to the forensic investigators case work as possible. The drawback is a lower degree of control of the material. For example we lack information on whether a hard disk is mechanical or solid-state drive (SSD) in some cases. This lack of information does not affect our results since we collect the data at the LBA level from the hard drive controller. The lower levels of physical storage formats are therefore hidden from us [10][57][5][14].

From our point of view the only difference between a mechanical hard disk and an SSD hard disk is their filling of unused areas, which can be either old data, 0x00 or 0xFF depending on how the TRIM command is implemented in the SSDs [9][31][32][33][30][29]. Hence a mechanical drive will more often give us old data from currently unallocated clusters than an SSD. Since we only use the LBA positions of unique data any 0x00 and 0xFF filling is automatically filtered out. In the case of old

data from unallocated clusters a very unbalanced erase/write cycle is required to leave a large amount of old data, i. e. first a large amount of data should be erased, followed by a small amount of (or no) writing of new data. This will be the case if a hard disk is erased using a random pattern and then reformatted and reused. If a large amount of the unallocated sectors contain old data, which are unique, they will have an effect on our results. To affect the map creation process to any greater extent the scenario shall be true for a significant part of the partitions in our data set. Before we collect the data we therefore check with the users if they have done any large file system cleaning close to our data collection.

The hard disks in our data set differ in size, ranging from 64 GB to 1 TB. We extract the largest NTFS formatted partition (in four cases there were an extra storage partition present which was extracted too) from each hard disk, based on the assumption that it contains the OS and user files. As can be seen in Table 1 the total size of the partitions in the experiment is 8 638.4 GiB, corresponding to 18 210 308 798 hashes. Of those 3 809 786 792 hashes are unique. The percentage of unique hashes for each partition is also shown in Table 1. A low number of unique hashes is an indication of the partition not being used, or at least not for storing user data. A low amount of unique hashes and a high amount of 0x00 or 0xFF filling can be seen in Table 1 for the bigger partitions (those ending in "b") of the hard drives where we used more than one partition to collect data.

The life time and hence amount of data stored on the hard disks vary. Most of the hard disks are filled with 0x00 to some extent. That can be remnants of the production process, but of the smaller hard disks ($\leq$ 256 GiB) some are SSD, which are filled with 0xFF from the factory [9]. To determine whether any of the partitions in our data set has been completely filled with data at any time during its life time we studied the last 20 GB of each partition. The size of 20 GB was chosen to be a suitable trade-off between a large enough amount of data and the risk of including the OS area for the smaller partitions. In Table 1 the partitions sizes and the amount of 0x00 and 0xFF filling are shown. A low amount of both 0x00 and 0xFF filling is an indication of the partition being (almost) completely filled with data during some stage of its life time. This could either be user data or a random data from a disk wiping tool.

We are only using partitions formatted as NTFS, because that is currently the most common file system among desktop systems having an approximate market share of 90% [49]. The partition names in Table 1 are given based on the order of hashing, i. e. partition "A" was hashed

Table 1. The sizes in GiB of the partitions in the experiment, their amount of unique hashes and 0x00 and 0xFF filling in the last 20 GB of the partitions. A low amount of filling is an indicator of the partition being completely filled or wiped with at random pattern at some stage during its life time.

Name	**Size [GiB]**	**Unique hashes [%]**	**0x00 fill [%]**	**0xFF fill [%]**
F	59.5	0.08	100.00	0
E	59.5	22.36	2.01	0.12
AC	111.3	7.63	100.00	0
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
K	146.4	5.82	100.00	0
Qa	150	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	86.85	0.02	0.02
Oa	209	13.68	100.00	0
Y	217.1	14.77	100.00	0
P	232.7	53.97	0.83	0.07
H	237.3	16.68	100.00	0
AA	237.3	12.60	100.00	0
D	237.9	20.59	0	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
Sb	265.6	36.22	100.00	0
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
X	443.8	6.48	100.00	0
U	448	10.60	100.00	0
V	465.6	48.43	100.00	0
Ob	699	0.15	98.72	0.00
Qb	766.5	1.47	100.00	0
B	905.2	29.74	100.00	0
Sum	8683.4	20.92	67.61	3.35

before "B" and so on. Four computers contain two partitions each that are included based on size (the computers were installed with an extra partition for user data). These partitions are indicated by a second lowercase letter in the name in Table 1. Although lacking an OS these partitions still contain an NTFS file system and therefore can be included in our data set.

The unique hash values we have found also include an amount of 1 KiB[3] MFT records. These records will result in up to two unique hash values each when hashing due to their highly varying content (time stamps, file names, file content etc). We therefore performed a survey on 27 computers not included in our data set estimating the mean number of MFT records by counting the total number of files and folders in the computers (since each file and folder in a computer is represented by, at least, one MFT record[4]). The result of the survey showed that the average total amount of files and folders in these computers were 363 630. Due to the uncertainty involved in the counting (we counted via the file explorer) the value includes an extra 25% added to cover for hidden files, files requiring more than one MFT record and any MFT records that are internal to the file system. The extra 25% also cover for any network storage of user data of the office grade computers in our file counting data set. In consumer grade computers all user files would probably have been stored locally and therefore included in our counting.

2.2 Implementation

To prepare the data for the experiment we extract and merge the hash data from the largest partitions into a single file, which is then sorted in ascending order of hash value. We then extract the unique hashes from the file, thus any 0x00 and 0xFF filled sectors are automatically filtered out. After the extraction of unique hashes we sort them in order of ascending LBA position and separated them in individual files based on partition identity. The data for each partition are then divided into 128 equally sized areas, each being $\frac{1}{128}$ of the size of the partition. Then we calculate the probability of finding unique hashes in each area through counting the number of unique hashes divided by the size of the areas in sectors for each partition.

After the probability calculation step we calculate the mean, median and standard deviation of the probability of unique hashes for each area of the partitions regardless of the differing partition sizes. The mean values are used as a map of a general storage media, showing where

it is more probable to find user data (unique data) in a generic NTFS formatted partition.

2.3 Evaluation

To evaluate our map we run an experiment simulating a hash-based carving scenario comparing the performance of sampling according to our map to a uniform sampling distribution. As ground truth we use four real partitions not included in our data set. We use the distribution of unique data in the four partitions to pick a random integer *target*. Then we use our map to pick a random integer *map* and the uniform distribution to pick a random integer *uni*. All random integers are selected within the same total range representing the LBA postitions of a fictive partition, although with bias for *target* and *map*. If $map = target$ our map gets one hit, if $uni = target$ the uniform distribution gets one hit. The predefined range is 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.

We iterate the random sampling process 10^9 times for each of the four partitions to stabilize the result. The low number of partitions used to create the map does however affect the evaluation since it is a small population to build a model from. Likewise our set of partitions forming the ground truth is small and the result is therefore affected by any individual variations of the partition content. Another factor affecting the result is the fact that the four partitions used as ground truth were taken from computers that should be scrapped and therefore had well used hard drives. They therefore contained a lower amount of 0x00 and 0xFF at the end.

The experiment was executed using Python 2.7 and the `random` library in a Debian Stretch (v. 9) computer.

3. Result

As can be seen in Figure 1 showing a map of our results the probabilities of unique hashes at different positions are varying between approximately 12% to 41%. The low median values in the second half of the partitions are due to the presence of 0x00 and 0xFF filling in a significant number of the partitions. If more than 50% of the partitions have no or a very low amount of unique data in that area the median value will be (close to) zero, which it is. The plot is based on splitting each partition into 128 blocks corresponding to $\frac{1}{128}$ of the partitions size.

When formatting a hard disk with NTFS 12.5% of the volume space is reserved for the MFT [48] as default. In all 30 partitions in our data

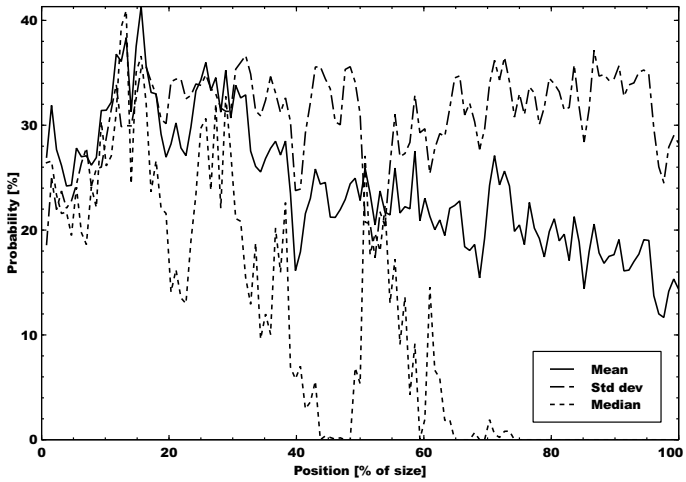

Figure 1. A plot of the mean, median and standard deviation of the probability of unique hashes (in percent) at different positions of the 30 partitions in our data set. The position is given as percent of the partition size. Each of the partitions is split into 128 equally sized areas based on the specific partition's total size. The behavior of the median plot in the second half is due to the low number of unique hashes in these parts of most of the partitions.

set the MFT area starts exactly 3 GiB into the partition. Hence the start of the area where non-resident file data are allocated can be found at LBA position $P = 3 * 2^{30} + 0.125 \cdot \textit{partition size in bytes}$, if not the user changes the MFT reserved space at the time of formatting. If the partition is very small the non-resident data allocation point is probably changed. Based on our data set the non-resident data allocation start is valid for partitions ≥ 60 GiB.

At the non-resident data allocation point the bulk of the OS, first user files and different software from the initial installation reside. The minimum space requirement for a Windows 7, 8, 8.1 and 10 installation is 20 GiB for 64-bit systems according to Microsoft [47][45][46]. The most probable start of storage of the day-to-day usage of a partition containing Windows is consequently at $(3 + 20) \cdot 2^{30} + 0.125 \cdot \textit{partition size in bytes}$ bytes into a partition. Transferring this to a percentage of the partition length gives in our case (see Figure 1) a value approximately between 14 and 43%. The highest amount of OS files is found in the beginning of the area and it decreases towards the end. This can explain the overall sharp negative trend of the plot between 20% and 40%, together with the peaks around 30%. Looking at the behaviour of the mean plot from 40% and upwards the values are slowly decreasing and the standard deviation is increasing. This is the effect of the differing usage patterns

of the partitions. Some have been storing more data, been more utilized, than other partitions in the data set.

We found 3 809 786 792 unique hashes in our data set, which correspond to data created locally by the user or the OS, such as logs. There are however unique parts of MFT records too in the data. Each file and directory is represented by at least one MFT record in NTFS[5]. The MFT records might affect the result by increasing the number of unique non-user data hashes. To estimate the effect from the MFT records we studied the number of files and folders in 27 typical computers (both office and home). We found the mean value to be 363 630 files, which corresponds to approximately 0.7% of the unique hashes in our 30 computers.

The `pagefile.sys` and `hiberfil.sys` might also generate a large amount of unique hashes depending on to what extent they are used. These files will certainly affect the map and our results, but since they are of high value to a digital forensic investigation they should be included in our data.

During the work with the mapping process we found four sectors containing the same hash value at the same LBA position in all partitions included in our data set. The sectors are found in file system cluster 786 435. They all contain the second half of MFT records which has only been used once according to their signature values [12, p. 352]. The first part of these MFT records contain similar, but not equal information. The `istat` tool [13] shows that the sectors belong to the `$MFT` file, i. e. the file system itself. The $DATA attribute of the $MFT files in the five computers we have raw access to all allocate the same eight clusters at the beginning of the run length (see Table 2). Combining this with the static content of the four sectors in cluster 786 435 the NTFS formatting seems to place the start of the MFT at the same position exactly 3 GiB into the partition. If this is true the first and last sectors of an NTFS partition should contain the hexadecimal string `00 00 0C 00 00 00 00 00` starting at position 0x30 [50] (little endian). We have verified this for the five computers we have raw access to.

According to Carrier the "$DATA attribute of the $MFTMirr file allocates clusters in the middle of the file system" [12, p. 303]. This implies that the middle sector, based on the size of the volume (the partition), is actually where the mirror should be kept. However, this is not always true. In four of the five computers we have full access to the `$MFTMirr` file allocates file system cluster 2 and in the last partition file system cluster 8 912 895 is allocated. However, the latter partition is 59 919 808 clusters in size, hence none of the `$MFTMirr` files are located

```
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. Part of the $DATA attribute of the $MFT file for five computers in our data set. The eight numbers on every third row indicate the file system clusters allocated to the file. File system cluster 786 435 contains the four static sectors (at positions 6 291 481, 6 291 483, 6 291 485 and 6 291 487) we have found in all 30 partitions.

near the middle of any of the partitions. Consequently the allocation strategy of NTFS seem to have changed since Carrier wrote his book.

To evaluate the efficiency of our map in random sampling situations we tested it against 4 NTFS partitions not included in the 30 used to create the map. Due to the low number of partitions used in the evaluation, the distribution of unique data in the individual partitions have a high impact on the result. We therefore regard the result as a first indicator of the performance of future maps, not the final answer. We are awaiting access to more data to be able to run a new evaluation. The result of the evaluation can be seen in Table 2.

The best result (when the map most resembles one of the evaluation partitions) is almost 10% better than using a uniformly distributed sampling rate. Varying the number of equally sized areas do not change the results in any significant way, neither do varying the fictive (16 MiB) partition size.

4. Discussion

Although the validity of the idea of looking for data where the probability of finding it is higher than randomly searching for data in a uniform

Table 2. The result of the evaluation of the map against four unrelated NTFS partitions, which are not included in the 30 partitions in the mapping data set. The table shows the number of hits using the map relative to using a uniformly distributed sampling rate. We used 16 MiB partitions divided into 128 equally sized areas sampled 10^9 times for the evaluation.

Map	**Uniform**	**Map/uniform** [%]
28635	30279	94.6
29881	30363	98.4
32556	30836	105.6
33257	30461	109.2
124329	121939	102.0

pattern is based on common sense we have also performed an empirical evaluation to test our specific implementation. The result shows an improvement of 2% when using our map compared to a uniformly distributed sampling rate. This might not create a paradigm shift, but it still is a positive indicator of the relevance of our idea. The reason for the seemingly poor result is the low number of partitions used to create the map. To reveal the underlying deterministic allocation pattern the amount of data needs to be much larger. Using a more solid statistical foundation will then improve the strength of the result. Having a big enough data set also allows us to divide it into several use cases, each one rendering its own map. The idea is to be able to diversify between for example web surfers, office administrators, file sharers. This however requires a much larger data collection effort, while maintaining a high level of control of the collected material to filter out unique data not created by the user or system, such as data written during disk wiping.

When the mapping foundation is stable there are several ways it can be used to improve the efficiency of the current digital forensic methods and tools, especially in file carving situations where there is no file system to be used. One example of usage is when using hash based carving to find parts of files in a hard disk. Then three different scenarios are possible:

Speed is prioritized. The total amount of samples is lowered compared to the uniformly distributed sampling case without any significant loss in detection ability. This scenario can for example be used in triage situations or when there is a need to get a preliminary answer quickly.

Speed is maintained. The same amount of samples are maintained compared to the uniformly distributed sampling case, which gives

a higher detection ability at the same execution speed. This is the standard case, which can be used without changing the digital forensic process.

Detection rate is prioritized. A larger amount of samples are used compared to the uniformly distributed sampling case, giving a much higher detection rate at a lower cost in execution speed. For example used in situations where the suspects hard disk has an unusual usage pattern. In this way the standard amount of hashes can be maintained in low priority areas and at the same time use a higher sampling rate for better detection ability in high priority areas of the hard drive.

When the area reserved for the MFT is used up a new area equalling 12.5% of the volume size is added. If possible that area is to be contiguous, but need not be. Hence as the file system grows new MFT records are added and allocated where suitable [12]. Thus an old and well used NTFS partition might 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 our 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 contain 510 zero bytes followed by a two byte long *signature value*[6] at the end of the sector. The worst case scenario is a partition filled with files less than approximately 700 bytes[7] in size, which would result in a partition filled with MFT records storing the data internally. If all files contained the same data only the MFT meta data (time stamps etcetera) would differ, thus the partition would still seem to be filled with random data. The maximum number of files in an NTFS partition is $2^{32} - 1$ [48], hence the partition would be approximately 4 TiB in size.

To estimate the amount of unique MFT record hashes in our data set in another way we 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 being unique in our data set represent a signature value of 3613 (0x1D0E, little endian). Many of the lower signature values generate several thousands of hits. There is however no guarantee that all the generated hashes belong to MFT records, but at least four do and consequently the amount of possibly unique MFT hash values polluting our data set is most prob-

ably less than 0.7%. Hence the unique hashes of the MFT records do not affect the precision of the map to any larger extent.

We have chosen to limit our experiment to computers running Microsoft Windows 7 and later and having NTFS formatted main partitions. To protect the privacy of the computer owners we use the cryptographic SHA-1 hash to obscure the real data. This limits our ability to trace the original data of each hash, but since we are only interested in the LBA position of unique hashes we do not need to know what data the hashes represent to be able to create a map.

Our work can also be used to find shared data. Of special interest is what we call static data, shared data that are found at the same LBA position in several unrelated storage media. Knowledge of the LBA position of static data will be of great use for a wide range of digital forensics applications. Together with for example forensic imaging and analysis prioritizing such knowledge can also provide an investigator with the means to break the encryption of a hard drive through a plain text attack [60], depending on the encryption algorithm used.

The LBA position of static data can be used to handle corrupt storage media. In many cases large parts of the corrupt media are readable, but there are no indications of the forensic value of the lost parts. Having access to a map of static content in storage media will help the digital forensic investigator to improve the evaluative reporting during case work by indicating the forensic value of any lost areas. This will in the end lead to a higher confidence in the collected evidence.

Furthermore a map can be used to create signatures to identify the correct file system in partially recovered partitions. Since the meta data layout and allocation process during installation are differing between OSs such signatures are feasible.

Finally areas that should have a high probability of static content, but do not, will work as an indicator of the presence of malware or any other suspicious activity in a file system, since deviations are unlikely in such areas. Instead of having to hash every file in a file system in search of deviations, the search can start at the most probable place in the file system. The partition is then scanned in descending order of probability of static content.

5. Conclusion and Future Work

Our work is based on the principle that it is better to search for something where the probability of finding it is higher. We therefore have developed a method to create a map of the probability of finding unique data at different LBA positions of storage media. The term

unique data is defined as data that are created locally on a computer and not (yet) shared. This includes both data created by the system, such as log files and data created locally by the user (not downloaded from the internet). Such uniquely created data are often more valuable to a forensic investigation than shared data, even though shared data of course can be valuable too.

The map provides the digital forensic investigator with a pre-calculated view of a generic storage media, which can be used to concentrate the forensic process on the relevant parts of the disputed material, instead of spending valuable time on first scanning the complete storage media from end to end. The concept of unique data is only used when creating the map, which is done once (apart from regular updates). When the map is finished it can be used repetitively for any data, method, tool or investigation process and without the need to recreate it for each new case.

The concept of creating a map of the probability of unique (or static) data at different positions of storage media opens up a new world of applications. It can for example be used in triage situations, when planning the order of analysis of large amounts of seized storage media, estimate the value of partially analyzed data due to corruption and for breaking encryption of storage media. We therefore plan to extend our data set to stabilize the map creation and make the map more reliable. We will also explore other methods to be used for creating maps, as well as the possibility to create maps for different use cases.

The four sectors with equal hash values that we found at approximately 3 GiB into all 30 partitions in our data set show that there might be specific areas of NTFS partitions that are static. We aim to search for and study the origin of any such areas as future work. We also plan to extend our approach to other file systems, especially ext4 and Apple File System (APFS), with the goal of creating a general mapping process for any storage media, regardless of type and file system.

We are releasing our current hash data set to the public, but due to its size the optimal transfer option will need to be agreed upon in each specific case. Please contact the first author to arrange for a transfer.

This work was sponsored by the Norwegian Research Council Ars-Forensica project number 248094/O70.

Notes

1. The theoretical risk of collisions come from the fact that 512 bytes of data are compressed into a 20 byte long hash and therefore the results might contain false positives. The problem can be viewed as a Birthday Paradox, where N is the number of possible hashes, n is the number of hashes, i. e. the total amount of sectors we have hashed (as a worst case

scenario), and $P(Collision)$ the probability of a collision, which can be calculated as

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

and with $N = 2^{160}$ and $n = 18\ 210\ 308\ 798$ the probability of at least one collision is approximately

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

Our approximation is based on Stirling's approximation of factorials, which gives acceptable results when dealing with very large numbers. Since the SHAttered [61][17] attack is 100 000 times faster than a brute force attack using the birthday paradox the risk of an intentional collision is higher, but the attack is unfeasible in our situation.

2. Windows 8 was introduced at the end of 2012 [27] and therefore cannot exist in the RDC, nor can Windows 10.

3. The size of an MFT record is defined in the boot sector of an NTFS partition. The de facto standard size is 1 KiB [12].

4. If a file has many attributes, for example alternate streams or is heavily fragmented, the file system creates a new MFT record to hold the extra information.

5. Depending on the number of attributes connected to a file more than one MFT record might be needed to store them. A typical example is a file with a lot of alternate data streams, or a highly fragmented file.

6. Signature values [12] are used by NTFS to verify the integrity of data structures (but not sectors containing file content) spanning two or more sectors. 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. These 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 sectors that are read have the same signature value, and thus belong to the same data structure. Every time a data structure is updated on disk the signature value is incremented by one [12].

7. The maximum size of an internal $Data attribute varies depending on the size 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 byte limit [48].

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 normalised compression distance as a file fragment classifier, *Digital Investigation*, 7(Supplement), pp. S24–S31, 2010.

[3] S. Axelsson, Using normalized compression distance for classifying file fragments, *2010 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: Ninth IFIP WG 11.9 International Conference on Digital Forensics, Orlando, FL, USA, January 28-30, 2013, Revised Selected Papers*, G. Peterson and S. Shenoi, (Eds.), pp. 171–182, 2013.

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

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

[7] F. Breitinger and K. Petrov, Reducing time cost in hashing operations, in *Advances in Digital Forensics IX: Ninth IFIP WG 11.9 International Conference on Digital Forensics, Orlando, FL, USA, January 28-30, 2013, Revised Selected Papers*, G. Peterson and S. Shenoi, (Eds.), pp. 101–117, 2013.

[8] F. Breitinger, C. Rathgeb and H. Baier, An Efficient Similarity Digests Database Lookup – A Logarithmic Divide & Conquer Approach, *Journal of Digital Forensics, Security and Law*, 9(2), pp. 155–166, 2014.

[9] C. Buckel, Understanding flash: Blocks, pages and program erases, (`flashdba.com/2014/06/20/understanding-flash-blocks-pages-and-program-erases/`), 2014.

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

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

[12] B. Carrier, *File System Forensic Analysis*, Addison-Wesley Professional/Pearson Education, Upper Saddle River, NY, USA, 2005.

[13] B. Carrier, Tsk tool overview, (`wiki.sleuthkit.org/index.php?title=TSK_Tool_Overview`), 2014.

[14] T.-S. Chung, D.-J. Park, S. Park, D.-H. Lee, S.-W. Lee and H.-J. Song, A survey of flash translation layer, *Journal of System Architecture*, 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, *2009 Fourty-Second Hawaii International Conference on System Sciences*, pp. 1–10, 2009.

[16] Digital Corpora, Real data corpus, (`digitalcorpora.org/corpora/disk-images/real-data-corpus`), 2018.

[17] Cryptology Group at Centrum Wiskunde & Informatica (CWI) and Google Research Security, Privacy and Anti-abuse Group, Shattered — we have broken SHA-1 in practice, (`shattered.io`), 2017.

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

[19] European Police Office (Europol), Internet organised crime threat assessment (IOCTA) 2016, Technical report, European Cybercrime Centre (EC3), The Hague, The Netherlands, 2016.

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

[21] K. Fairbanks, A technique for measuring data persistence using the ext4 file system journal, *2015 IEEE Thirty-Ninth Annual Computer Software and Applications Conference*, vol. 3, pp. 18–23, 2015.

[22] K. Fairbanks, An analysis of ext4 for digital forensics, *Digital Investigation*, 9(Supplement), pp. S118–S130, 2012.

[23] S. Fitzgerald, G. Mathews, C. Morris and O. Zhulyn, Using NLP techniques for file fragment classification, *Digital Investigation*, 9(Supplement), 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, Naval Postgraduate School, Monterey, California, USA, 2012.

[25] 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*, 7(Supplement), pp. S13–S23, 2010.

[26] S. Garfinkel and M. McCarrin, Hash-based carving: Searching media for complete files and file fragments with sector hashing and hashdb, *Digital Investigation*, 14(Supplement 1), pp. S95–S105, 2015.

[27] S. Gibbs, From Windows 1 to Windows 10: 29 years of Windows evolution, *The Guardian*, (`www.theguardian.com/technology/2014/oct/02/from-windows-1-to-windows-10-29-years-of-windows-evolution`), 2014.

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

[29] Y. Gubanov and O. Afonin, Why SSD drives destroy court evidence and what can be done about it, (`belkasoft.com/download/info/SSD%20Forensics%202012.pdf`), Belkasoft LLC, Palo Alto, CA, USA, 2012.

[30] Y. Gubanov and O. Afonin, *Recovering evidence from SSD drives in 2014: Understanding trim, garbage collection and exclusions*, (`articles.forensicfocus.com/2014/09/23/recovering-`

evidence-from-ssd-drives-in-2014-understanding-trim-garbage-collection-and-exclusions/), 2014.

[31] Y. Gubanov and O. Afonin, *SSD and eMMC forensics 2016, part 1*, (articles.forensicfocus.com/2016/04/20/ssd-and-emmc-forensics-2016/), 2016.

[32] Y. Gubanov and O. Afonin, *SSD and eMMC forensics 2016, part 2*, (articles.forensicfocus.com/2016/05/04/ssd-and-emmc-forensics-2016-part-2/), 2016.

[33] Y. Gubanov and O. Afonin, *SSD and eMMC forensics 2016, part 3*, (articles.forensicfocus.com/2016/06/07/ssd-and-emmc-forensics-2016-part-3/), 2016.

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

[35] M. Karresand, *Completing the Picture — Fragments and Back Again*, Licentiate thesis, Linkping Institute of Technology, Linkping University, Linkping, Sweden, 2008.

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

[37] 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, ACSF*, pp. 11–20, 2006.

[38] M. Karresand and N. Shahmehri, Oscar – file type identification of binary data in disk clusters and RAM pages, *Proceedings of IFIP International Information Security Conference: Security and Privacy in Dynamic Environments (SEC2006)*, pp. 413–424, 2006.

[39] M. Karresand and N. Shahmehri, Oscar – using byte pairs to find file type and camera make of data fragments, *Proceedings of the Second European Conference on Computer Network Defence, in conjunction with the First Workshop on Digital Forensics and Incident Analysis (EC2ND 2006)*, pp. 85–94, 2007.

[40] M. Karresand and N. Shahmehri, Reassembly of fragmented jpeg images containing restart markers, *Proceedings - Fourth Annual European Conference on Computer Network Defense, EC2ND 2008*, pp. 25–32, 2008.

[41] S. Key, *File block hash map analysis*, (`www.guidancesoftware.com/app/File-Block-Hash-Map-Analysis`), 2012.

[42] J. Kornblum, Identifying almost identical files using context triggered piecewise hashing, *Digital Investigation*, 3(Supplement), 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 (SAISMC 2010)*, pp. 236–247, 2010.

[44] Microsoft, *Default cluster size for NTFS, FAT and exFAT*, (`support.microsoft.com/en-us/help/140365/default-cluster-size-for-ntfs--fat--and-exfat`), 2015.

[45] Microsoft, *System requirements*, (`support.microsoft.com/en-gb/help/12660/windows-8-system-requirements`), 2017.

[46] Microsoft, *Windows 10 system requirements*, (`support.microsoft.com/en-us/help/4028142/windows-windows-10-system-requirements`), 2017.

[47] Microsoft, *Windows 7 system requirements*, (`support.microsoft.com/en-us/help/10737/windows-7-system-requirements`), 2017.

[48] Microsoft, *How NTFS works*, (`technet.microsoft.com/pt-pt/library/cc781134(v=ws.10).aspx`), 2018.

[49] Net Applications.com, *Desktop operating system market share*, (`www.netmarketshare.com/operating-system-market-share.aspx?qprid=10&qpcustomd=0`), 2017.

[50] NTFS.com, *NTFS partition boot sector*, (`www.ntfs.com/ntfs-partition-boot-sector.htm`), 2018.

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

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

[53] R. Poisel, M. Rybnicek and S. Tjoa, Taxonomy of data fragment classification techniques, in *Digital Forensics and Cyber Crime: Fifth International Conference, ICDF2C 2013, Moscow, Russia, September 26-27, 2013, Revised Selected Papers*, P. Gladyshev, A. Marrington and I. Baggili, (Eds.), pp. 67–85, Springer International Publishing, Cham, Switzerland, 2014.

[54] D. Quick and K.-K. Choo, Data reduction and data mining framework for digital forensic evidence: Storage, intelligence, review and

archive, *Trends & Issues in Crime and Criminal Justice*, 480, pp. 1–11, 2014.

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

[56] D. Quick and K.-K. Choo, Big forensic data reduction: digital forensic images and electronic evidence, *Cluster Computing*, 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, No. 8898371, (`www.freepatentsonline.com/8898371.html`), 2014.

[58] V. Roussev, Managing terabyte-scale investigations with similarity digests, in *Advances in Digital Forensics VIII: Eighth IFIP WG 11.9 International Conference on Digital Forensics, Pretoria, South Africa, January 3-5, 2012, Revised Selected Papers*, G. Peterson and S. Shenoi (Eds.), Springer, Berlin, Germany, pp. 19–34, 2012.

[59] N. Rowe, Identifying forensically uninteresting files using a large corpus, in *Digital Forensics and Cyber Crime: Fifth International Conference, ICDF2C 2013, Moscow, Russia, September 26-27, 2013, Revised Selected Papers*, P. Gladyshev, A. Marrington and I. Baggili (Eds.), Springer International Publishing, Cham, Switzerland, pp. 86–101, 2014.

[60] B. Schneier, *Applied Cryptography – Protocols, Algorithms, and Source Code in C*, 2 edition, John Wiley & Sons Inc., Hoboken, NJ, USA, 1996.

[61] M. Stevens, E. Bursztein, P. Karpman, A. Albertini and Y. Markov, The first collision for full SHA-1, *Advances in Cryptology – CRYPTO 2017*, pp. 570–596, 2017.

[62] A. Tridgell, *Spamsum README*, (`www.samba.org/ftp/unpacked/junkcode/spamsum/README`), 2002.

[63] R. van Baar, H. van Beek and E. van Eijk, Digital forensics as a service: A game changer, *Digital Investigation*, 11(Supplement), 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*, 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, 2007 (IAS 2007)*, pp. 393–398, 2007.

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