DFRWS APAC 2021 Author Preprint

A Contemporary Investigation of NTFS File Fragmentation

Vincent van der Meer[a,b,d], Hugo Jonker[b,c], Jeroen van den Bos[d]

[a]*Zuyd University of Applied Sciences, vincent.vandermeer@zuyd.nl*
[b]*Open University of the Netherlands, hugo.jonker@ou.nl*
[c]*Radboud University Nijmegen*
[d]*Netherlands Forensic Institute, j.van.den.bos@nfi.nl*

Abstract

There is a significant amount of research in digital forensics into analyzing file fragments or reconstructing fragmented data. At the same time, there are no recent measurements of fragmentation on current, in-use computer systems. To close this gap, we have analyzed file fragmentation from a corpus of 220 privately owned Windows laptops.

We provide a detailed report of our findings. This includes contemporary fragmentation rates for a wide variety of image-, video-, office-, database-, and archive-related extensions. Our data substantiates the earlier finding that fragments for a significant portion of fragmented files are stored out-of-order. We define metrics to measure the degree of "out-of-orderedness" and find that the average degree of out-of-orderedness is non-negligible. Finally, we find that there is a significant group of fragmented files for which reconstruction is insufficiently addressed by current tooling.

Keywords: File Fragmentation, File Carving, Digital Forensics

1. Introduction

File fragmentation impacts (amongst others) file system performance and file recovery. Indeed, many studies in these areas rely on assumptions with respect to fragmentation. In the domain of digital forensics, this includes studies into file fragment classification (e.g., Rahmat et al. [13]), generic file carvers (e.g., Ying and Thing [16], Garfinkel [5]) as well as file type specific file carvers (e.g., Durmus et al. [4], Yang et al. [15]), and fragment dating (e.g., Bahjat and Jones [2]).

For all such studies, contemporary data on file fragmentation is a necessary prerequisite to determine carving strategies. The most recent large-scale study of file fragmentation is from 2007 by Garfinkel [5], with data gathered from 1998 to 2006. This corpus is now outdated: it concerns mostly FAT-type file systems, while Windows (since XP) by default uses the NTFS file system. Moreover, this corpus concerns deprecated versions of Windows whose combined market share is below 1.75%[1].

To remedy this, we gathered data of file fragmentation on NTFS file systems from 220 laptops. These machines are individually acquired, owned and maintained, and are in regular use by their owners. As these machines were owned by volunteer participants, privacy was paramount. Therefore, we designed a privacy-friendly approach to data gathering [10]. In that work, we also presented initial fragmentation findings. Key amongst those was that out-of-order fragmentation occurs fairly frequently – a type of fragmentation that seems to mostly have been overlooked in literature.

Contributions. In this paper, we present in-depth, contemporary data on NTFS file fragmentation. The main contributions are:

- Our corpus provides a contemporary (Oct'18 – Jan'19) view on file fragmentation.
- The number of files in the corpus is significantly larger (2–10 times) than previous works (>1 mln `.jpg`; 14,000 `.doc`; 87,000 `.docx`; ...).
- We provide novel metrics on the convolutedness of fragmentation: *degree of internal fragmentation* and *degree of out-of-orderedness*.
- We report on a number of fragmentation characteristics: fragmentation vs. file size, fragmentation vs. used volume space, fragmentation per extension, gap size for files fragmented in two parts, distribution of number of fragments, correlation between fragmentation and disk size, fragmentation and disk type (primary / secondary).
- We find (amongst others) that the average degree of out-of-orderedness of fragmented files is non-negligible. This has implications for the field of digital forensics.

2. Background

2.1. Terminology

We make use of the following NTFS terminology:

MFT: Master File Table; contains metadata (including allocated blocks) for all files.

[1] https://netmarketshare.com/operating-system-market-share.aspx?id=platformsDesktopVersions

DFRWS APAC 2021 Author Preprint

Resident files: Files without allocated blocks, whose data is stored completely in its MFT record.

Compressed files: Files may be compressed by NTFS itself, as opposed to application-level compression. This compression is transparent to any application using NTFS.

Sparse files: Files where only blocks containing non-zero data are stored. The file size of sparse files is thus typically larger than allocated on disk. (Used e.g. for virtual machine files.)

Hard links: An MFT entry may contain more than one path + filename. These names appear to the user as individual files, but there is only one physical representation on disk.

Symbolic links: A symbolic link is an MFT entry that points to a path + filename (possibly on another volume, including non-NTFS volumes). They thus contain no data, only meta data.

Volume: A storage device is a physical unit for storing data. It is partitioned into one or more volumes, which in Windows are addressable via drive letters.

In addition, we use HDD to denote *Hard Disk Drive*, i.e., a storage medium based on magnetic storage with moving read and write heads and spinning discs; and SSD to denote *Solid State Drive*, i.e., a storage medium based storing data based on integrated circuits (typically flash memory), without moving parts. With respect to the popularity of SSDs versus HDDs: in our dataset (Table 7), we find that 84% of the laptops use an SSD, and 67% uses an HDD.

2.2. Data storage and deletion on SSDs

SSD devices operate differently than HDDs. For example, to extend the longevity of the disk, they typically use wear leveling: a technique to avoid writing overly much in one area of the disk. However, wear leveling happens in the firmware and is thus invisible to the NTFS file system. That is: it does not affect the operation of NTFS, and the NTFS file system is not aware of this taking place.

SSD devices also handle deletion differently than HDDs. Regular HDDs handle deletion by marking the deleted blocks of the disc as available. That is, regular HDDs leave the data on the disc until it is overwritten. In contrast, an SSD drive cannot write to an already occupied part. Thus, each block must be empty before it can be written to. The earliest SSDs used a form of garbage collection to empty deleted blocks. This matured into the creation of the TRIM command, which wipes the specified blocks. Once blocks have been wiped, their data is physically removed from the disc and thus the data no longer recoverable. This raises the question of whether recovery of deleted files is possible at all on SSDs.

Nisbet et al. [12] show that once the TRIM command has been sent to the drive, erasing usually takes places within minutes. They also show that, within the time frame of deleting a file by the user and the execution of the TRIM command, significant amounts of data can still be recovered, with small files being fully recovered, and for large files being partially recovered. After the execution of the TRIM command however, only up to 0.6% of the data was recoverable. This places concrete boundaries on the forensic effectiveness for file carving.

In case the data on the SSD was subjected to a successfully executed TRIM command, the data thus is not realistically recoverable. However, there is not a one-to-one correspondence between file deletion and successful execution of a TRIM command. In particular, there are various reasons why a SSD either is not TRIM-enabled, or that a TRIM command is not successfully executed[2].

2.3. Fragmentation

The NTFS file system stores files into blocks, where each block occupies a fixed size on disk. Blocks are identified by their block number. A file is thus assigned a list of block numbers. A file is *not* fragmented if the assigned block numbers are listed in order, and these block numbers are consecutive. When this is not the case, the file is fragmented. This may be because the blocks occur out of order, because the block numbers are not consecutive, or both. This gives rise to four storage patterns, as depicted in Figure 1. Of these storage patterns, in-order contiguously stored files are not fragmented. The other patterns describe fragmented files.

	Contiguous	*Non-contiguous*
In-order	A B	A . . . B
Out-of-Order	B A	B . . . A

Figure 1: Examples of the four storage patterns for a bi-fragmented file

Two types of fragmentation can occur on a file system:

1. *fragmentation of free space* is caused due to the deletion and shrinking of files. While these operations typically do not fragment the file itself, they do create unallocated space that is likely not adjacent to the (other) already existing unallocated space.
2. *file fragmentation* occurs when the file system does not write a file contiguously. File fragmentation can happen when new files are created or existing files are extended. Note that file system implementations may choose to do so even when it is not strictly necessary (i.e., when there is sufficient contiguous free space available).

We refer to the various parts of a fragmented file as fragments. More specifically, a file consists of a number of blocks, which are grouped into one or more *fragments*. A fragment is contiguous and in-order, and cannot be extended with more blocks of the same file while remaining in-order and contiguous.

[2] https://www.forensicfocus.com/articles/recovering-evidence-from-ssd-drives-in-2014-understanding-trim-garbage-collection-and-exclusions

2.4. Degree of fragmentation

The degree of fragmentation can be defined in various ways, depending on what is considered the total number of files. In literature, it is not always clear which definition is used. It is the ratio of the number of fragmented files divided by a total. Different choices can be made for the total, which gives rise to four definitions.

definition 1 (degree of fragmentation). *The degree of fragmentation is the number of fragmented files divided by the total number of files. The total number of files defined as:*

I. *all MFT entries, OR*
II. *all MFT entries with data, OR*
III. *all MFT entries with blocks assigned, OR*
IV. *all MFT entries with ≥ 2 blocks assigned.*

Note that definition I covers all MFT entries, including symbolic links; definition II excludes symbolic links; and definition III furthermore excludes resident files. Nevertheless, definition III still includes files of one block – which inherently cannot fragment. Definition IV is the only definition which excludes all non-fragmentable MFT entries from consideration. It is thus the most strict, while definition I is the most broad definition (gives the smallest degree of fragmentation).

We consider definition IV most relevant for reporting on measurements of file fragmentation. Definition III is useful when the number of blocks of a file is unknown (e.g., in file carving). Most studies unfortunately do not clarify which definition they use.

3. Related Work

There have been three large-scale studies reporting on file fragmentation. We summarise their findings in Table 1.

source	*year*	*% frag*	*used frag. definition*
[5] Garfinkel	2007		
– *all file systems*		6	?
– *NTFS file systems*		12.2	?
[11] Meyer & Bolosky	2012	4	?
[10] Van der Meer et al.	2019	2.2	all MFT entries (I)
		4.4	fragmentable files (IV)

Table 1: Comparison of fragmentation rates found in literature

The seminal large-scale study into file fragmentation is due to Garfinkel [5]. He gathered data from over 300 used hard disks. The data set includes 219 FAT file systems, 51 NTFS file systems and 5 UFS file systems. He found an average percentage of file fragmentation of 6%. Most findings are reported over the entire data set. The paper does provide sufficient information to derive the fragmentation rate over all 51 NTFS file systems, namely 12.2%.

Garfinkel reports several findings. He found different file types have different fragmentation rates, that most fragmented files are split into two parts (bifragmented), and he reports on the gap size between the two fragments of bifragmented files. It is not clear which definition of degree of fragmentation Garfinkel uses in his paper.

In a study on file system content of 597 Windows computers, Meyer and Bolosky [11] reported finding a level of file fragmentation of 4%. In addition, the most highly fragmented files within their data set were log files. Note that it is not clear which definition Meyer and Bolosky used to calculate the percentage of fragmented files.

We previously [10] reported on the same data set we analyse in this paper. There, we focus on how to perform data acquisition in a privacy-friendly manner, providing only scant data on fragmentation findings. We found that over 46% of fragmented files were fragmented out-of-order. To the best of our knowledge, this is the first report of out-of-order fragmentation found in practice. Finally, we reported that in comparison to previous studies the percentage of fragmented MFT entries has reduced, yet the absolute amount of fragmented data has increased.

While the previous works focused on desktops and laptops, several studies have investigated file fragmentation on smartphones. Ji, Chang, Shi, Wu, Li and Xue [8] report on EXT4 fragmentation behaviour on four Android smartphones. They observe that files, especially database files, may suffer from severe fragmentation. In a follow-up study using five smartphones, Ji, Chang, Hahn, Pan, Shi, Kim, Xue et al. [9] find that, under daily use, fragmentation quickly begins to occur. They find that for such devices, fragmentation is strongly correlated with disk space utilization. Moreover, the specific way how SQLite files are used (frequent deletions, synchronous writes) exacerbates fragmentation as well.

Finally, Darnowski and Chojnacki [3] derive a model of NTFS block allocation algorithms that predicts how a new file will be stored. They propose modelling the NTFS allocation strategy as a finite state machine. They define a sequential model for writing files, which provides predictions on block allocation. These predictions include predicting when fragmentation occurs and even cover out-of-order fragmentation. They confirm the accuracy of their model via synthetic experiments.

4. Data collection and processing

Data was collected from the personal machines of volunteer student participants, between October 2018 and January 2019. The machines were individually bought, managed, and maintained by their respective owners. The student population is divided into classes. By visiting each class once, we ensured no double participation. Data was collected by a custom-made privacy-friendly data gathering tool based on Fiwalk, by Garfinkel [6]. The output of this is standardised DFXML[3] structured data. This was converted into an SQLite database for analysis.

The data set consists of input from 220 laptops. Three of these ran Windows 7, the other 217 ran Windows 10 (four of which in a dual-boot configuration). With respect to storage devices configurations: 111 laptops contained an SSD + an HDD,

[3] https://forensicswiki.xyz/wiki/index.php?title=Category:Digital_Forensics_XML

	NTFS-compressed	*sparse files*	*hardlinks*	*resident files*	*symbolic links*	*MFT entries [10]*
all	598,119	242,844	8,778,592	12,639,771	1,380,728	84,390,537
with data	597,255	97,322	8,659,294	12,616,364	–	82,960,039
with blocks	597,255	97,322	8,079,067	–	–	70,320,268
with ≥ 2 blocks	367,284	75,645	5,365,324	–	–	42,671,054
fragmented files	72,351	24,079	34,720	–	–	1,871,109
out-of-order frag. files	40,660	12,156	14,259	–	–	868,917
% fragmented						
of all	12.1 %	9.9 %	0.4 %	–	–	2.2 %
of those with data	12.1 %	24.7 %	0.4 %	–	–	2.3 %
of those with blocks	12.1 %	24.7 %	0.4 %	–	–	2.7 %
of those with ≥ 2 blocks	19.7 %	31.8 %	0.6 %	–	–	4.4 %
of fragmented files:						
out-of-order fragmented	56.2 %	50.5 %	41.1 %	–	–	46.4 %
avg. internal fragmentation	8.6 %	12.8 %	24.3 %	–	–	*19.9 %*
avg. OoO'ness	32.5 %	29.2 %	24.1 %	–	–	*29.9 %*

Table 2: Fragmentation per MFT entry type. For the right-most column, *italicized text* presents new additions in comparison to [10].

70 contained a single SSD, 36 contained a single HDD, and 3 laptops contained a dual SSD configuration. On six storage devices, one or more volumes were encrypted and thus not accessible for data collection. In total, these 334 storage devices contained 733 volumes: 729 NTFS, and 4 EXT4. We exclude the EXT4 volumes from consideration.

Of the NTFS volumes, 707 volumes had a block size of 4096 bytes. Other NTFS block sizes were rare: 14 volumes had a block size of 512 bytes; 7 had a block size of 1024 bytes and 1 volume had a block size of 2048 bytes.

5. Results

In this section, we present our results. Note that many of the distributions on which we report are skewed. To provide some insight into the skewedness, we present both average and median values for such distributions.

The results will be presented using the different definitions on fragmentation (primarily def. I and def. IV), where we use the most relevant definition of the degree of fragmentation per context. However, these metrics do not convey how complex the fragmentation of a file is. Two aspects determine the complexity of a file's fragmentation: the number of fragments (relative to the file size) and the order between the fragments. To provide insight into the complexity of fragmentation, we introduce two corresponding metrics: the *percentage of internal fragmentation* to quantify the number of fragments in relation to the file size, and the *percentage of out-of-order'ness* (OoO'ness for short), which quantifies the extent to which the fragments occur out of order. Both definitions make use of the number of fragmentation points, which is the number of times a process reading the file sequentially would need to jump over one or more blocks to continue reading the file.

definition 2 (% of internal fragmentation). *The percentage of internal fragmentation of a file f of at least 2 blocks is the ratio of the number of fragmentation points vs. the number of blocks minus one, i.e.:*

$$\text{intfrag}(f) = \frac{\text{fragpoints}(f)}{\text{blocks}(f) - 1} \cdot 100,$$

where blocks(f) *denotes the total number of blocks of file f, and* fragpoints(f) *is the number of times where, when reading a block of file f, the next block of f is not the next block on disk.*

For example, a file f_1 whose blocks are stored contiguous and in order has 0 fragmentation points and therefore intfrag(f_1) = 0%. Another example, consider a file f_2 of N blocks, where the blocks occur in order, but every block of f_2 is followed by a block of another file. In this case, there is a fragmentation point after every block except the last block of the file. Thus, fragpoints(f_2) = $N - 1$, which gives intfrag(f_2) = $\frac{N-1}{N-1} \cdot 100$ = 100%.

definition 3 (% of OoO'ness). *The percentage of out-of-order'ness of a fragmented file f is the ratio of the number of times the next fragment occurs prior to the current vs. the total number of fragmentation points, i.e.:*

$$\text{OoOness}(f) = \frac{\text{backfragpoints}(f)}{\text{fragpoints}(f)} \cdot 100,$$

with fragpoints(f) *defined as before, and where* backfragpoints(f) *denotes the number of times the next block of file f is stored earlier on disk than the current block.*

For example, consider a file f_3, of N blocks, which is contiguous, but written backwards. I.e., the second block is the block before the first block; the third block is the block before

				% of fragmented files with ... fragments:						of fragmented files:		
				in-order			out of order					
ext	# files with ≥2 blocks	% fragmented def. I	def. IV	2	3	≥4	2	3	≥4	avg. % intfrag	avg. % OoO'ness	avg. # fragments
Image												
bmp	70,425	1.6	2.5	40.7	9.7	5.4	14.6	10.7	18.8	10.3	29.2	3.2
gif	276,241	0.8	1.8	40.4	7.9	5.3	10.3	9.0	27.0	28.4	26.4	3.6
jpeg	13,774	8.5	8.7	24.8	8.8	6.0	10.6	9.1	40.7	13.6	33.6	3.8
jpg	1,043,198	2.7	3.1	32.6	6.2	3.8	13.8	10.5	33.1	12.4	33.5	4.4
png	2,389,752	0.9	3.1	48.9	9.5	3.5	14.4	10.2	13.4	32.9	25.6	2.8
psd	7,022	4.5	4.5	31.0	7.8	1.6	16.6	17.2	25.7	6.8	37.2	9.3
psp	422	4.6	6.2	15.4	15.4	11.5	3.8	7.7	46.2	5.7	24.8	8.7
raw	5,246	1.1	1.2	57.8	4.7	0.0	12.5	18.8	6.3	3.5	24.9	17.1
tif	6,309	9.3	9.7	13.3	18.7	5.7	3.6	19.7	39.0	4.8	31.3	4.1
Video												
avi	9,800	1.8	1.8	9.6	1.7	2.8	0.0	1.1	84.7	1.1	29.9	40.8
flv	332	26.8	26.8	6.7	3.4	1.1	1.1	4.5	83.1	1.5	38.5	29.8
mkv	2,404	2.7	3.1	44.0	2.7	0.0	12.0	8.0	33.3	0.1	32.4	6.8
mov	4,459	4.3	4.4	30.9	0.5	0.5	13.9	13.9	40.2	0.5	39.0	20.2
mp4	38,007	6.4	6.5	31.3	5.7	2.5	14.4	11.2	35.0	1.0	36.7	28.8
mpg	3,269	0.4	0.4	0.0	15.4	0.0	7.7	0.0	76.9	1.9	42.0	21.8
mts	1,591	0.2	0.2	33.3	0.0	0.0	0.0	33.3	33.3	0.0	52.4	4.3
wmv	27,328	0.7	0.7	33.2	3.6	0.0	35.7	10.7	16.8	1.8	50.1	5.8
Office												
doc	14,831	5.1	5.5	21.4	9.7	9.7	8.5	13.4	37.3	15.8	31.4	5.1
docx	87,077	6.0	6.2	35.1	8.6	5.6	13.5	9.7	27.4	16.5	30.2	4.6
msg	7,120	0.7	6.2	75.7	0.0	0.0	23.6	0.0	0.7	38.2	24.1	2.1
odt	2,147	4.8	4.9	44.8	6.7	5.7	23.8	7.6	11.4	35.9	33.7	2.8
pdf	92,117	7.9	8.1	14.6	6.1	9.7	6.7	8.9	53.9	7.3	33.6	9.3
ppt	3,406	7.9	8.0	7.0	0.0	3.3	5.1	1.5	83.1	3.0	37.4	10.9
pptx	17,846	11.6	11.7	8.7	2.5	6.3	5.5	4.2	72.8	3.4	36.3	19.2
prf	1,113	0.9	4.6	66.7	0.0	0.0	23.5	0.0	9.8	31.6	26.8	2.4
pst	120	33.1	35.8	55.8	2.3	0.0	9.3	23.3	9.3	0.0	24.7	2.8
rtf	80,977	0.9	1.0	39.4	3.5	6.1	29.9	9.7	11.4	6.7	40.7	3.5
xls	8,550	2.0	2.3	22.2	5.7	8.2	5.2	13.9	44.8	13.9	33.8	5.2
xlsx	17,721	4.1	4.1	48.6	12.3	3.8	16.6	8.6	10.1	27.3	30.4	3.3
Database												
accdb	1,450	12.0	12.0	8.6	3.4	2.9	2.9	13.8	68.4	4.7	40.6	30.0
db	33,320	12.0	17.4	28.2	7.8	3.5	8.8	9.2	42.5	19.5	32.2	24.5
mdb	11,052	3.8	6.1	21.1	7.9	4.0	14.0	13.5	39.5	9.8	39.2	5.1
sqlite	7,959	26.2	27.8	44.3	5.6	2.2	20.5	7.1	20.4	9.0	33.2	6.9
Archive												
7z	3,568	12.2	18.1	58.5	7.7	1.9	6.7	9.9	15.3	49.7	19.0	31.8
gz	48,900	1.8	3.7	33.4	21.0	6.2	5.9	12.5	20.9	56.2	20.5	6.4
rar	3,589	7.3	7.5	13.7	4.8	2.6	5.2	7.4	66.3	3.5	34.5	48.1
zip	53,919	7.9	11.2	22.9	7.6	7.9	8.5	7.7	45.5	15.9	30.4	22.4

Table 3: Fragmentation per extension (categorised)

the second, etc. In this case, every fragmentation point is backwards, hence OoOness(f_3) = 100%. In contrast, OoOness(f_2) = 0%, as file f_2 was stored in-order, so backfragpoints(f_2) = 0.

Remark that extreme values of OoO'ness correspond to relatively simple cases: an OoO'ness of 100% is a file where the next block is always stored earlier on disk (e.g., f_3), and an OoO'ness of 0% concerns a file where the next block is always stored further (e.g., f_1). In contrast, an OoO'ness of 50% means half the fragmentation points are backwards – i.e., when reaching the end of a fragment, there is no preference for either forward or backward direction to find the next block. Thus, an average OoO'ness of 50% is a worst-case (with respect to out-of-orderedness) situation for a file carver.

5.1. Fragmentation per MFT entry type

In Table 2, the main fragmentation characteristics of our data set are presented, split per MFT entry type. For completeness and comparison purposes, we include our previously [10] reported totals (right column), extended with new measures of average internal fragmentation and average OoO'ness. Remark that both resident files and symbolic links can inherently not fragment. In our data set, we find that hard-linked files are up to 7 times less likely to be fragmented than the average. Sparse files and NTFS compressed files were already known to be prone to fragmenting; to the best of our knowledge, we are the first to quantify the extent of this. In the data set, we find that (under definition I) around 10% of both sparse and NTFS compressed files are fragmented. Under a stricter definition of fragmentation, one that only considers files that may

potentially fragment (i.e., files with at least two blocks), the ratios increase to one in five (NTFS compressed) and close to one in three (sparse), respectively. Finally, note that when NTFS-compressed files are fragmented, the average degree of internal fragmentation is lower than average (8.6% vs. 19.9%).

5.2. Fragmentation per file extension

Table 3 provides various data on the fragmentation per extension. In this table, we list the number of files with at least 2 blocks (i.e., the number of files relevant for definition IV), as well as the percentage of files that are fragmented. Specifically, we include both def. I for comparison purposes, and def. IV as most representative definition of fragmentation. Furthermore, like Garfinkel [5], we provide the percentage of fragmented files that are fragmented into 2, 3, and 4 or more parts completely in-order, and similar for files that are fragmented at least partially out of order. For the fragmented files, we also provide the average internal fragmentation (definition 2), the average OoO'ness (definition 3), as well as the average number of fragments.

Images. Fragmented images are often fragmented out of order. For fragmented `bmp`, `png`, and `raw` files, the percentage of fragmented files that are fragmented out-of-order are 44.1%, 38.0% and 37.6%, respectively. For all other image formats, fragmented files are more likely to be fragmented out-of-order than in-order.

Videos. Yang et al. [15] claim `avi` files are more likely to be fragmented than other files. Our dataset does not corroborate this. We find that the average fragmentation rate for `avi` files (1.8%) is lower than the general average (4.4%). However, when `avi` files are fragmented, the number of fragments is often large (average of 40.8 fragments).

The `.mts` format is a video format typically used in camcorders. In our dataset, 2 systems account for 1,555 of the 1,591 `mts` files.

Office documents. Interestingly, Outlook `pst` files are often fragmented (35.8%). The number of fragments is low, leading to a negligible rate of internal fragmentation. The main complexity in recovering fragmented `pst` files is due out-of-orderedness. Another interesting document-related finding is that `pdf` files have a higher fragmentation rate than the word-processing extensions `rtf` (Wordpad), `odt` (OpenOffice), `doc` and `docx` (MS Word); an unexpected result considering `pdf` files are typically static, i.e., not intended for editing.

Databases. Ji et al. [9] studied fragmentation on Android systems and found that database files are prone to fragmentation, due to concurrent and frequent growth. Our dataset shows that this is true on NTFS systems as well: all database extensions are fragmented above average.

File size†	*# fragmented files*	*% frag*
min⋆ – 10 kB	11,531,201	1.8
10– 50 kB	15,669,438	3.9
50 – 100 kB	4,468,221	4.9
100 – 500 kB	6,490,196	7.4
0.5 – 1 MB	1,573,008	6.8
1 – 5 MB	2,096,812	8.2
5 – 10 MB	397,872	7.8
10 – 50 MB	341,782	9.5
50 – 100 MB	45,148	14.0
100 – 500 MB	44,534	21.4
> 500 MB	12,842	46.1

⋆ min: 2 assigned blocks, irrespective of file size and block size.
† kB = 1,000 bytes, MB = 1,000,000 bytes.

Table 4: Fragmented files per file size

File size†	*% OoO*	*% intfrag*	*% OoO'ness*	*# fragments*	
				avg	*median*
min⋆ – 10 kB	18.7	77.0	18.2	2.0	2
10– 50 kB	29.7	26.0	24.9	2.3	2
50 – 100 kB	47.1	10.7	32.2	2.8	2
100 – 500 kB	57.5	5.5	34.3	3.6	3
0.5 – 1 MB	70.2	2.9	37.3	5.5	4
1 – 5 MB	76.2	2.2	38.1	10.0	5
5 – 10 MB	80.4	1.5	37.4	23.1	7
10 – 50 MB	82.6	1.3	37.2	49.2	12
50 – 100 MB	76.7	1.3	33.7	126.3	14
100 – 500 MB	66.2	0.6	35.9	156.8	3
> 500 MB	74.1	0.1	36.3	93.1	4

⋆ min: 2 assigned blocks, irrespective of file size and block size.
† kB = 1,000 bytes, MB = 1,000,000 bytes.

Table 5: Fragmentation characteristics of fragmented files versus file size

5.3. Fragmentation in relation to file size

Tables 4 and 5 show fragmentation and fragment properties split out in file size intervals. The ranges include start point, and exclude the end point. With regards to the smallest file that may be fragmented: this is dependent on the number of allocated blocks. Note that allocated blocks do not need to be filled. Indeed, we found 10 fragmented files, whose file size was 1 byte.

Table 4 shows that smaller files occur more often than larger files. Note that 74% of all files of at least two blocks are smaller than 100 kB. Furthermore, we make the following observations:

- Of all fragmented files with a file size between 1 and 100 MB, over 75% is fragmented out-of-order.
- As file size increases, the number of fragments typically increases (though this correlation is not perfect).
- For files >50 kB, the average OoO'ness is slightly over a third, more or less irrespective of the file size. This means that at each fragment boundary, there is, on average, a probability of about $\frac{1}{3}$ that the next fragment is located before the current fragment, and a probability of about $\frac{2}{3}$ of the next fragment being ahead.
- We found that some files are extremely fragmented, such as one file split into 20,000 fragments. This skews the

average, but the median value of the range is less affected and provides a more nuanced view on the number of fragments.

5.4. Distribution of the number of fragments

In Table 6, we extend our previously reported fragmentation data [10, pg. 5, Table II] with file size and gap size information.

As we reported previously, 56.76% of files is bi-fragmented (fragmented into two parts). In-order bi-fragmented files are common amongst fragmented files, they constitute 41.84% of all fragmented files. Theoretically, as files are fragmented into more parts, it is increasingly less likely that all fragments occur in order. Our data set corroborates this.

Finally, note that the average OoO'ness is hardly correlated with the number of fragments. For any file fragmented into three or more fragments, average OoO'ness is yet again roughly a third.

5.5. Gapsize distribution of bi-fragmented files

For in-order fragmented files, the gap between two consecutive fragments is unambiguously defined as the distance from the last block ("tail") of the first, to the first block ("head") of the second. For out-of-order files, there is not one, unique, unambiguous definition of the distance between two consecutive fragments. Note that since Garfinkel's study does not consider out-of-order fragmented files, a direct comparison is not possible.

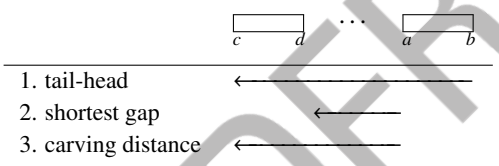

Figure 2: Possible metrics for gapsize of OoO fragmented files

Figure 2 depicts three possible metrics. All three metrics have their applications. The first, tail-head distance, covers the total length to be covered, but includes the length of both fragments themselves. For file carving, this is not that useful: once the first fragment is found, this will be skipped when searching for further fragments. The second metric, shortest gap distance, measures the shortest distance between the two fragments, which only makes sense if both fragments are known. The third metric, carving distance, measures the distance an out-of-order file carver would have to make. This includes the fragment length of the unknown fragment, but skips the already-found fragment.

Note that when looking at in-order fragmented files, these three metrics are equivalent. It is only when the next fragment appears before the current fragment that differences arise.

Figure 3 depicts the number of in-order bi-fragmented files with a distance of 1 to 300 blocks. The part shown in the figure covers 10.0% of all distances between the fragments of in-order bi-fragmented files. The large trends depicted in the figure hold over the entire range; in particular, we found that distances in

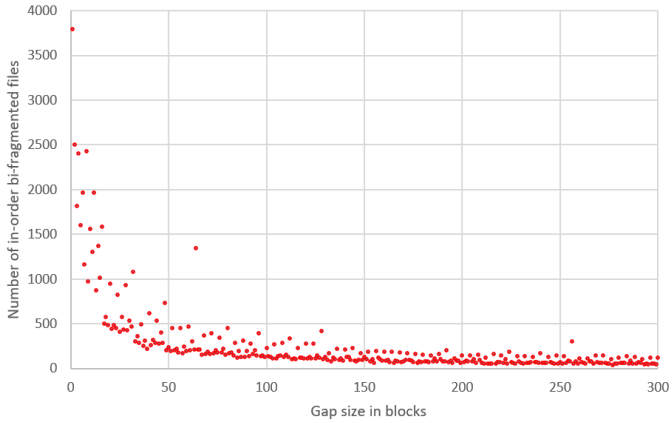

Figure 3: Gap-size distribution of in-order 2-fragmented files

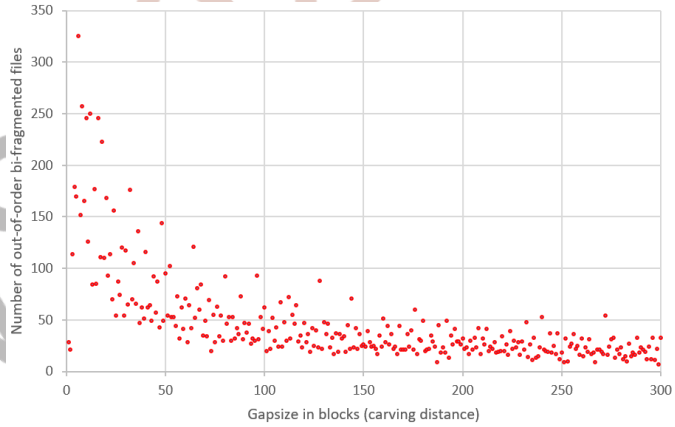

Figure 4: Gap-size distribution of out-of-order 2-fragmented files

general decline, with a generic exception for gapsize distances that are a power of two (see also Table B.11 in the appendix).

We evaluated all three distances for out-of-order bi-fragmented files. We found that there are only small deviations between them. Interestingly, the peaks at distances of powers of two as seen for in-order files occurred much more strongly for carving distance than for the other two distance metrics. Hence, from here on out we will use this metric for the gapsize of out-of-order files.

Figure 4 depicts the carving distances for out-of-order files. As was the case for in-order files, the main trends depicted in the figure continue across the entire range. The gapsizes depicted in the figure cover 5.0% of all out-of-order bi-fragmented files.

Lastly, concerning the aforementioned preference for gap lengths of powers of two: note that these gap lengths are not necessarily aligned with specific locations on disk. More specifically, the length of the first fragment determines the gap start. This preference for gap lengths of powers of two thus seems to be an artefact of how NTFS assigns blocks. Consequently, the fact that carving distance aligns well with these observations suggests that carving distance aligns with how NTFS allocates blocks.

#fragments	#files	% OoO	average OoO'ness	Sum of all gap sizes (in blocks) (carving-distance) min	average	median	max
2	1,062,539	26.3	26.3%	1	7,038,401	711,673	517,861,056
3	340,422	56.5	32.8%	2	15,594,100	5,406,252	990,207,960
4	160,472	74.9	34.9%	3	25,645,754	11,833,174	811,066,168
5	93,835	84.6	35.5%	4	34,104,018	16,748,220	969,975,568
6–10	122,388	91.5	36.7%	5	53,979,693	28,003,492	2,002,994,256
11–20	45,031	93.5	36.8%	11	90,567,873	47,230,761	3,037,661,708
21–100	35,890	92.4	36.8%	42	227,973,821	96,196,576	9,852,412,280
101–1000	9,721	93.6	34.6%	399	1,194,774,735	345,806,721	69,129,433,312
1001+	811	96.4	30.3%	17,636	5,760,340,498	948,806,352	270,488,355,485

Table 6: Distribution of number of fragments per file

5.6. Percentage of used volume space and file fragmentation

As a volume becomes more filled with data, the remaining unallocated space becomes progressively more scarce and more likely to be fragmented. This may impact for the degree of fragmentation. For example, Ji et al. [9] concluded from their study of Android devices that the degree of fragmentation is highly correlated with the percentage of used volume space.

We examined this in our data set. First of all, we excluded volumes with very few files (≤ 15), as we do not consider such volumes to be in active daily use (but act e.g., as recovery partition). Moreover, they contain so few files, that even a single fragmentation on such a volume will strongly skew the fragmentation rate, and thus, strongly affect the correlation. For example, in our data set there are 44 volumes that each contain 3 files, one of which is fragmented (i.e., a fragmentation percentage of 33%).

Given these constraints, we find a moderate positive relation between data fragmentation and the percentage of used volume space. For SDDs we find that the correlation is 0.462, and for HDDs the correlation is 0.464. Though the correlation coefficients are nearly identical, the underlying data distribution is rather different, as shown in Figure 5.

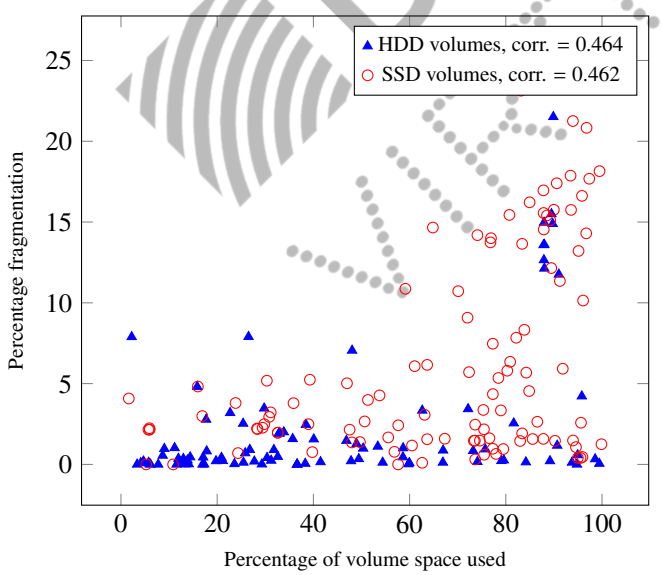

Figure 5: Fragmentation vs. used volume space

5.7. Fragmentation per storage device

For non-dual-boot systems, we distinguished between primary (boot disk) and secondary storage devices within our data set based on file count and extension occurrence. This is possible as a Windows install has roughly 80,000 files, with many system-related extensions such as `.dll` and `.com`. For every non-dual-boot system in our data set, these heuristics provided a clear division between primary and secondary storage device.

By default, Windows has a scheduled defragment-task, with different schedules for SSDs (monthly) and HDDs (weekly). The defragmentation strategy can differ per storage device[4].

Table 7 shows that single disk SSD-systems are more fragmented than single disk HDD-systems, on average 2.4 times more. The most common system configuration is a SSD/HDD combination. In this configuration, the primary SSDs are way more fragmented than secondary HDDs, on average 5.2 times more. Note that in this data set there was no system with a dual HDD configuration.

Storage Device	#	average frag	median frag
Single disk (SSD)	67	5.6 %	2.0 %
Single disk (HDD)	36	2.3 %	1.2 %
Primary disk (SSD)	113	7.3 %	4.7 %
Secundary disk (HDD)	110	1.4 %	0.2 %
Secundary disk (SSD)	3	4.1 %	3.9 %

Table 7: Fragmentation per storage device

5.8. Other extremes and curiosa

- In our data set, there are 2,914 file extensions for which no file happened to be fragmented. The top 10 most occurring of these is listed in Table 8.
- Among the extremely fragmented files (files with thousands fragments or more), the most frequent occurring extensions are `exe`, `log`, `xml`, `dat`, and `dll`.
- Of all the 1,871,109 fragmented files, only 8 are fragmented contiguous out-of-order. All these 8 files are bi-fragmented.

[4] https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/defrag

- Some files occupy vastly more blocks than their file size requires. One file in our corpus had a 1 byte file size, yet had 369 blocks allocated on disk. Moreover, this 1 byte file was fragmented (out-of-order) into 5 fragments.

			File size in bytes		
Ext	*# files (def. IV)*	*# systems*	*Avg.*	*Median*	*St. Dev.*
`ctt`	51,315	28	7,596	4,904	5,545
`ovl`	47,998	14	313,040	33,599	3,716,227
`p7x`	36,030	99	10,693	10,653	733
`tt`	24,032	47	27,017	26,786	3,466
`anm`	23,807	12	42,248	22,522	125,422
`vcd`	21,288	8	1,959	1,547	4,509
`prx`	20,885	197	11,889	4,286	241,195
`slp`	20,461	7	926,198	63,459	3,236,241
`p7s`	17,252	42	9,834	9,355	2,036
`ovs`	16,310	8	3,735,952	51,213	8,313,296

Table 8: Top 10 most frequently occurring extensions without fragmented files

	# files reported		*% fragmented*	
file type	2007, [5]	2020, def. IV	2007	2020
Image				
`bmp`	26,018	70,425	8	2.5
`gif`	357,713	276,241	8	1.8
`jpeg`	108,539	13,775	16	8.7
`jpg`	–	1,043,198	–	3.1
`png`	9,995	2,389,752	5	3.1
Office				
`doc`	7,673	14,831	17	5.5
`ppt`	1,120	3,406	8	8.0
`pst`	70	120	58	35.8
`xls`	2,159	8,550	11	2.3
Video				
`avi`	998	9,800	20	1.8
`mpeg`	168	9	17	11.1
`mpg`	–	3,269	–	0.4
Database				
`mdb`	402	11,052	27	6.1

Table 9: Comparison of fragmentation rates between 2007 and this paper

6. Discussion

6.1. Overall fragmentation rates

The overall fragmentation rates (Sec. 5.1) have implications for file carvers. First, an upside for file recovery tooling: most files are not fragmented. This means that file recovery tools which ignore fragmentation (which are far easier to construct) will recover most files. Indeed, various studies assume that files are not fragmented, such as Gladyshev and James [7], Sportiello and Zanero [14].

However, there is also a downside: out-of-order files constitute close to half of all fragmented files. This means that any tool that aims to recover fragmented files, *must* account for out-of-order fragmentation. This impacts existing studies. For example, neither the file carver due to Garfinkel [5] nor the file carver for fragmented `jpg` files due to Abdullah et al. [1] account for out-of-order fragmentation.

6.2. Fragmentation per extension

The general trend of less fragmentation compared to previous studies extends also to specific files. In Table 9, we compare our findings to those reported in the 2007 study by Garfinkel. Note that `jpeg` and `jpg` file formats are equivalent, but they use a different extension. The same holds true for the `mpeg` and `mpg` file format. In Table 9, we compare Garfinkel's findings against our ratio as determined by def. IV (fragmentable files). We find a lower fragmentation rate across all extensions.

6.3. Implications for file carving

Although file fragmentation is a topic that attracts some interest in the digital forensics research community, most popular file carvers used in practice focus almost exclusively on recovering unfragmented files. This is an understandable choice given the considerable time it takes to carve large disks and other media even in the simplest scenarios.

This paper makes it possible for developers and users of file carvers to make informed choices about the type of recovery they implement and use. It allows an assessment of the added benefits of actually using bifragment gapcarving and whether to extend such an algorithm to include out-of-order fragments or extend it to something else, such as reconstructing files containing multiple gaps.

An important contribution is the explicit measurement of the incidence of out-of-order fragmented files (Tables 2, 3, 5 and 6), especially given that this is a large (percentage-wise) subset of all fragmented files. Additionally, the reporting on encountered actual gap sizes (Sec. 5.5) allows for practical estimations of the performance impact on deploying such an extended file carver. Given the amount of data fragmented out of order, as reported in this paper, the impact of a file carver able to reconstruct such files can now be properly ascertained.

6.4. Carving of NTFS-compressed and sparse files

NTFS allows special storage modes that do not store the actual file contents as-is on disk: NTFS-compression and sparse files. For both types, the blocks as stored on disk are not sufficient to reconstitute a file. Note that either mode may be used irrespective of a file's contents or file type. Thus, these NTFS storage modes could pose a challenge for file carvers.

Yoo, Park, Lim, Bang and Lee [17] state that most file carvers are unable to handle NTFS-compressed data (irrespective of fragmentation). They consider files of at least one block. In our dataset, only 0.8% of all files with allocated blocks (597,255 / 70,320,268) is NTFS-compressed. Yoo et al. propose a file carver to recover NTFS-compressed files. Their carver does not account for fragmented NTFS-compressed files, which (in our dataset) constitutes 12.1% of all NTFS-compressed files with blocks. Interestingly, their carver is targeted at NTFS-compressed `avi`, `wav` and `mp3` files. In our data set, the percentage of these files that are NTFS-compressed is 0.1%, 0.0%

and 0.1%, respectively (Table A.10).

With respect to sparse files, we find only three extensions (of those investigated) have a significant portion of them as sparse: `pst` (12.8%), `sqlite` (9.6%), and `db` (7.7%). All of these are significantly more fragmented than the average: 35.8%, 27.8% and 17.4%, respectively. The percentage of sparse files for the other studied extensions remains below 0.5%.

7. Conclusions

We performed a contemporary study into file fragmentation. Our data set is comprised of disk information from 220 personally acquired, owned, and managed machines. The data was collected in a period of 4 months (Oct'18 – Jan'19).

Previous reports lacked a clear definition on which files were considered. We remedied this by distinguishing four possible definitions of fragmentation rates, from including all MFT entries to only including MFT entries that could possibly fragment. We focused our reporting on the latter definition: files that could possibly fragment. We found an average fragmentation rate of 4.4%, which presents a significant decrease compared to Garfinkel's 2007 study. This decrease is also evident on the level of individual file types.

We reported on a number of fragmentation characteristics, including the convolutedness of fragmented files and the gapsize. To assess the convolutedness of fragmented files, we proposed two novel metrics: *degree of internal fragmentation* and *degree of out-of-orderedness*. Fragments are separated by a gap. We noted that there are three possible definitions of gapsize in case the next fragment precedes the current. Although the differences between these definitions are not very large, the *carving distance* still stood out: of the three, its measurements most strongly showed the "powers-of-two" gapsize property that forward-measured gapsizes so strongly exhibit.

Future work. We intend to design and implement a modern file carver supporting in- and out-of-order fragmentation. Furthermore, carving of fragmented NTFS-compressed files and carving of sparse files is currently unexplored territory. We found that sparse files mostly concern system-related file extensions. We are not aware of any file carver tailored for recovering sparse files, and we are exploring ways to implement such a file carver.

Acknowledgements. The authors would like to thank Guy Dols for his technical support, and all the volunteers (device owners) for their collaboration in this research. Van der Meer was supported by the Netherlands Organisation for Scientific Research (NWO) through Doctoral Grant for Teachers number 023.012.047.

References

[1] Abdullah, N.A., Ibrahim, R., Mohamad, K.M., Hamid, N.A., 2013. Carving linearly JPEG images using unique hex patterns (UHP), in: Proc. 1st Conference on Advanced Data and Information Engineering (DaEng'13), Springer. pp. 291–300.

[2] Bahjat, A.A., Jones, J., 2019. Deleted file fragment dating by analysis of allocated neighbors. Digital Investigation 28, S60–S67.

[3] Darnowski, F., Chojnacki, A., 2018. Writing and deleting files on hard drives with NTFS. Computer Science and Mathematical Modelling 8, 5–15.

[4] Durmus, E., Korus, P., Memon, N.D., 2019. Every shred helps: Assembling evidence from orphaned JPEG fragments. IEEE Trans. Information Forensics and Security 14, 2372–2386.

[5] Garfinkel, S.L., 2007. Carving contiguous and fragmented files with fast object validation. Digital Investigation 4, 2–12.

[6] Garfinkel, S.L., 2009. Automating disk forensic processing with sleuthkit, XML and python, in: Proc. 4th IEEE Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE'09), pp. 73–84.

[7] Gladyshev, P., James, J.I., 2017. Decision-theoretic file carving. Digital Investigation 22, 46–61.

[8] Ji, C., Chang, L., Shi, L., Wu, C., Li, Q., Xue, C.J., 2016. An empirical study of file-system fragmentation in mobile storage systems, in: Proc. 8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage'16), pp. 1–5.

[9] Ji, C., Chang, L.P., Hahn, S.S., Pan, R., Shi, L., Kim, J., Xue, C.J., et al., 2018. File fragmentation in mobile devices: measurement, evaluation, and treatment. IEEE Transactions on Mobile Computing .

[10] van der Meer, V., Jonker, H., Dols, G., van Beek, H., van den Bos, J., van Eekelen, M., 2019. File fragmentation in the wild: a privacy-friendly approach, in: Proc. 11th IEEE Workshop on Information Forensics and Security (WIFS'19), IEEE. pp. 1–6.

[11] Meyer, D.T., Bolosky, W.J., 2012. A study of practical deduplication. Transactions on Storage 7, 14:1–14:20.

[12] Nisbet, A., Lawrence, S., Ruff, M., 2013. A forensic analysis and comparison of solid state drive data retention with trim enabled file systems, in: Proc. 11th Australian Digital Forensics Conference (ADFC'13), SRI Security Research Institute. pp. 1–10.

[13] Rahmat, R., Nicholas, F., Purnamawati, S., Sitompul, O., 2017. File type identification of file fragments using longest common subsequence (lcs), in: International Conference on Computing and Applied Informatics (2016), IOP Publishing. pp. 1–9.

[14] Sportiello, L., Zanero, S., 2012. Context-based file block classification, in: Proc. 8th IFIP WG 11.9 International Conference on Digital Forensics, Springer. pp. 67–82.

[15] Yang, Y., Xu, Z., Liu, L., Sun, G., 2017. A security carving approach for AVI video based on frame size and index. Multimedia Tools Appl. 76, 3293–3312.

[16] Ying, H., Thing, V.L.L., 2010. A novel inequality-based fragmented file carving technique, in: Proc. 3rd International Conference on Forensics in Telecommunications (ICST'10), Springer. pp. 28–39.

[17] Yoo, B., Park, J., Lim, S., Bang, J., Lee, S., 2012. A study on multimedia file carving method. Multimedia Tools and Applications - MTA 61, 1–19.

Appendix A. Auxiliary data per extension

In Table A.10, we provide auxiliary data on file sizes. The right-hand side of this table focuses on NTFS-compressed and sparse files. Recovery of such files is complex, irrespective of whether they are fragmented or not. Therefore, results concerning these file types in Table A.10 are reported on all files with blocks (def. III), and not only files that could fragment (def. IV).

Appendix B. Gapsizes of powers of two

Table B.11, shows the frequency of gap sizes (in blocks) of powers of 2 for bi-fragmented files in our dataset. For comparison, we also show the incidence for adjacent gapsizes.

	# systems		file size in bytes using def. IV:				using def. III:		
ext	*def. III*	*def. IV*	*avg.*	*median*	*st. dev.*	*max.*	*# files*	*% NTFS-compressed*	*% sparse*
Images									
bmp	214	213	380,653	36,176	4,603,975	1,150,221,432	105,371	0.1	0.0
gif	214	214	73,113	14,878	587,069	67,859,584	468,406	0.4	0.0
jpeg	181	177	430,900	132,696	921,712	19,905,785	14,137	5.1	0.1
jpg	215	215	469,789	43,499	1,383,941	202,187,275	1,157,750	1.7	0.1
png	215	215	77,909	13,385	819,778	443,815,127	6,551,794	0.6	0.0
psd	156	156	6,098,161	318,875	25,696,786	657,852,455	7,111	1.1	0.0
psp	68	67	2,913,965	164,864	13,461,247	79,354,648	444	5.0	0.0
raw	207	200	4,943,742	38,144	22,211,217	340,245,502	6,038	0.0	0.0
tif	188	178	2,174,293	178,288	9,322,608	536,980,180	6,512	0.8	0.0
Videos									
avi	199	199	20,822,230	730,952	105,051,447	1,886,142,464	9,805	0.1	0.4
flv	34	34	25,612,283	3,670,220	83,269,828	911,348,494	332	0.0	0.0
mkv	206	206	250,121,103	109,239,726	351,135,887	1,994,939,880	2,406	0.1	0.2
mov	95	95	51,779,439	15,788,157	119,573,812	1,925,087,760	4,478	1.6	0.0
mp4	214	214	55,803,648	2,005,846	211,305,187	1,998,753,571	38,155	2.4	0.1
mpg	92	92	2,802,244	569,095	33,224,606	1,644,236,800	3,269	0.0	0.0
mts	8	7	168,426,360	113,362,944	191,510,208	1,893,931,008	1,747	0.0	0.0
wmv	206	206	4,187,920	398,973	47,347,382	1,892,176,290	27,382	0.0	0.0
Office									
doc	214	214	480,189	43,520	9,447,633	1,000,000,000	15,666	0.7	0.0
docx	214	214	371,838	29,359	2,032,033	117,328,214	87,124	2.8	0.0
msg	213	213	28,180	4,823	111,153	4,486,144	36,296	0.7	0.0
odt	140	140	152,304	16,500	818,144	24,663,942	2,147	1.1	0.0
pdf	215	215	2,619,014	462,167	12,769,129	695,725,963	93,265	1.1	0.0
ppt	210	210	1,462,596	802,816	2,396,460	35,269,926	3,406	0.6	0.0
pptx	211	211	4,711,035	1,089,065	16,722,112	871,334,541	17,851	1.3	0.1
prf	119	118	15,741	8,405	101,644	3,145,728	3,156	0.2	0.0
pst	31	27	152,551,442	173,720,576	213,757,822	1,896,784,896	125	2.4	12.8
rtf	214	214	183,797	82,239	1,051,434	77,456,537	90,604	0.3	0.1
xls	207	207	252,638	67,072	642,497	15,325,184	9,891	0.1	0.0
xlsx	214	214	205,915	17,573	4,159,236	307,409,090	17,729	1.1	0.0
Databases									
accdb	190	190	2,029,463	724,992	6,719,223	145,084,416	1,450	2.1	0.0
db	215	215	4,075,278	74,752	54,485,792	1,988,837,638	41,762	1.5	7.7
mdb	175	175	233,414	31,773	764,419	18,874,368	14,762	5.6	0.2
sqlite	212	212	782,992	65,536	7,712,433	454,340,608	8,245	1.8	9.6
Archives									
7z	201	201	37,437,492	112,778	143,243,604	1,926,983,279	5,170	0.7	0.0
gz	213	213	243,467	10,277	7,615,496	816,336,896	84,665	1.7	0.0
rar	161	161	49,886,452	5,883,486	156,451,000	1,927,419,308	3,667	0.4	0.1
zip	217	217	18,503,962	168,076	102,922,058	1,988,366,193	67,884	0.3	0.0

Table A.10: Meta information per extension (categorised)

Gap	*# Files*	*Gap*	*# Files*	*Gap*	*# Files*	*Gap*	*# Files*	*Gap*	*# Files*	*Gap*	*# Files*
2^0	3793										
2^1	2507	...	...	...	...	...	...	...	...	...	...
3	1819	15	1016	63	212	255	77	1023	28	4095	10
2^2	2408	2^4	1587	2^6	1346	2^8	300	2^{10}	78	2^{12}	20
5	1603	17	498	65	210	257	57	1025	23	4097	4
...	...	...	...	...	...	...	...	...	...	...	...
7	1165	31	469	127	121	511	39	2047	23	8191	3
2^3	2431	2^5	1083	2^7	421	2^9	138	2^{11}	43	2^{13}	7
9	972	33	302	129	106	513	46	2049	11	8193	2

Table B.11: In-order bi-fragmented gap sizes around powers of 2.