Forensic Science International: Digital Investigation 42-43 (2022) 301476

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

It is about time—Do exFAT implementations handle timestamps correctly?

Rune Nordvik [a, b, *], Stefan Axelsson [a, c]

[a] Norwegian University of Science and Technology, Norway
[b] Norwegian Police University College, Norway
[c] DSV, Stockholm University, Sweden

ARTICLE INFO

Article history:
Received 22 June 2022
Accepted 17 October 2022
Available online 29 October 2022

Keywords:
Digital Forensics
Timezone
Timestamps
Metadata
File systems

ABSTRACT

Digital forensic investigations require that file metadata are interpreted correctly. In this paper we focus on the timestamps of the exFAT file system. How these timestamps are written may depend on the implementation of the file system. We have performed experiments using Windows, MacOS and Linux to examine whether the respective file system drivers for exFAT use timestamps in the same manner, and whether they take the directory entry *UTCOffset* fields into account. We have also studied whether the forensic tools: Autopsy, X-Ways Forensics, EnCase Examiner, and FTK Imager interpret the timestamps consistently.

The results show that there are substantial inconsistencies both in the file system implementations and in how forensic tools handle these inconsistencies. For the unwary forensic examiner, there is a clear risk of interpreting timestamps incorrectly by a substantial margin.

We conclude that timestamp interpretation during criminal investigations should not be based on the assumption that the file system specifications are followed flawlessly by the file system driver developers or necessarily interpreted and displayed correctly by the digital forensic tools.

1. Introduction

During the investigation of criminal cases it is important that timestamps are interpreted correctly. Misinterpreting timestamps may exclude the guilty or implicate the innocent. For instance; a Word document on a USB stick belonging to the suspect with a creation timestamp corresponding to the time the crime was committed may indicate that the suspect was using a computer at the time of the crime to store the file on the USB stick. If traces found on the suspect's home computer also show that the same USB stick was inserted 1 h before the time of the crime, and removed a day after, this may indicate that someone was at the suspect's home at the creation of the document. If the crime took place at another address at the same time, this finding may exclude the person as a suspect if the investigation can connect the suspect to the computer. If further hypothesis testing shows that the computer has not been connected to any other network than the home network, and that the computer clock has not been manipulated, this will strengthen the main hypotheses that the suspect or someone else was at the suspect's home address at the time of the crime. However, more detailed hypothesis testing must be performed before this conclusion is firmly drawn.

The above deductions can only be drawn if file systems follow their specifications. However, these may not be available or only partly available. Specifications may also be misinterpreted by the implementer. For example, a previous study of N-version programming by Knight and Leveson (1986) show that developers tend to make similar mistakes even when they are following the same specification, i.e. the flaws they introduce in the code are not independent of each other. This means that there is little support to trust a tool is correct only by comparing its results with a similar tool (Nordvik et al., 2021).

In this paper we focus on the exFAT file system. The specifications are available from Microsoft (2021b). The exFAT file system can be used on Windows, MacOS, Linux, and other operating systems (Bretel, 2017). Off the shelf removable storage devices are today often pre-formatted with exFAT as it can work on most computers and supports file system volume sizes much larger than

* Corresponding author. Norwegian University of Science and Technology, Norway.
E-mail address: rune.nordvik@phs.no (R. Nordvik).

https://doi.org/10.1016/j.fsidi.2022.301476

the 32 GiB FAT32 limitation (Microsoft, 2021a). These removable devices can be mounted with read and write support on all previous mentioned operating systems, all thanks to the usage of the exFAT file system (USB Memory Direct, 2022).

For most users of file systems it is not critical that timestamps are accurate. However, when investigating criminal cases the accuracy of times given could be critical in order to answer the **when** question in the 5WH questions (Ieong, 2006). Ieong (2006) describes these 5WH questions as "What (the data attributes), Why (the motivation), How (the procedures), Who (the people), Where (the location) and When (the time)".

When it comes to the investigation of metadata, such as timestamps from file systems, most investigations take for granted that the digital forensic tools are able to parse the file systems that they claim to support, and courts depend on the tools accuracy and reliability (Nordvik et al., 2021). As timestamp interpretations may give a suspect an alibi, the investigation should not rely solely on tool interpretation.

Law enforcement organisations typically have a large backlog of seized devices waiting for acquisition and analysis (Scanlon, 2016). Digital forensic investigators use digital forensic tools when investigating criminal cases to increase the efficiency of the investigation. The concept of trust in criminal investigations is discussed by Neale et al. (2022), including erroneous trust in the accuracy of digital forensic tools. Error rates for digital forensic tools are seldom available (Nordvik et al., 2021), and previous research have shown that it is difficult to measure error rates involving all independent variables (conditions) that may impact the dependent variable (here the error rate) (Lyle, 2010).

In order to comply with certain human rights, like the right to privacy, family life, and correspondence (Article 8) (Court of Human Rights, 2021), law enforcement may utilise a more granular acquisition of files, e.g. by only including files from a specific time range (European Committee for Standardization, 2022). In these cases law enforcement of course depend on accurate timestamp interpretations.

Therefore, one can hardly overemphasise the importance of manual verification of tool findings, and how file system timestamps are interpreted.

The contributions of this paper:

- File system driver developers do not implement exFAT equally.
- When specifications are made available, they are not necessarily followed.
- Digital forensic tools have a tendency to make assumptions about metadata.
- Even when specifications are available, reverse engineering by performing black box testing is necessary.

To the best of our knowledge, it has not been performed experiments including multiple operating systems and multiple driver implementations of the same exFAT file system. Since digital forensic investigators do not necessarily know which operating system a removable device has been connected to, they should not assume that the exFAT specifications were followed by the driver. Even building digital forensic tools to automate file system parsing require a detailed understanding about how the file system drivers store metadata, and this paper will show that it is not necessarily always the case.

1.1. Background

The background information presented in this section is based on Microsoft (2021b). The exFAT file system has a volume boot record (VBR) which contains information necessary to find the important metadata structures such as the file allocation table (FAT), the cluster heap (data region), and the cluster of the Root directory. It also defines the size of a sector, cluster, the size of the volume, the number of FATs, the length of each FAT, and percentage of allocated clusters in the cluster heap.

The data region starts at cluster 2 and it is recommended that exFAT implementations place the root directory after the clusters used for the allocation bitmap and the up-case table. The allocation bitmap defines the allocation status of all clusters in the data region. The allocation bitmap has a corresponding set of directory entries, and its primary directory entry has the type 0×81. The number of allocation bitmaps correspond with the number of FATs, and this is normally 1, or maximum of 2.

The exFAT FAT table is mainly used for fragmented files, which are files that do not use contiguous clusters. When a file becomes fragmented, the stream extension directory entry will set the *NoFatChain* field to zero, meaning the FAT is in use. By using the *FirstCluster* field in the stream extension directory entry, the system identifies the correct cluster start in the FAT, and it can continue to the next cluster in the allocation chain by reading FAT chain. This enables the system to find all fragmented clusters for the file. If the file is not fragmented, the FAT is not in use, by setting the *NoFatChain* to one. Then the system can use the *FirstCluster* and the *DataLength* fields to extract the contiguous clusters for the file.

The root directory contains files and folders and each of them have a set of directory entries, as shown in Fig. 1. Allocated files have a file directory entry (type 0×85), a stream extension directory entry (type 0xC0), and one or more file name directory entries (type 0xC1), which is a set of allocated directory entries as illustrated in the top part of Fig. 2. The file directory entry contains timestamps, *UTCOffset* fields, file attributes, and a directory entry set checksum. The stream extension directory entry gives information about where the data content (the stream) is stored (FirstCluster, DataLength), the length of the FileName, and a hash of the FileName in upper case. The file name directory entry describes the name of a file, and will need NameLength/15 file name directory entries.

In this paper we mainly focus on the timestamps found in file directory entries, as shown in Table 1. There are three different timestamps in a file directory entry (FDE); Create, Last Modified, and Last Accessed. According to the specifications, to interpret the actual time the timestamp, the 10 ms increments, and the UTC offset should be considered (Microsoft, 2021b). The UTC offset describes the offset from UTC to local time (including daylight savings adjustments) in 15 min increments. The 10 ms increments are only available for the Create and the Last Modified timestamp, and increases the granularity from 2 s to 10 ms.

In Fig. 3 we see the *UTCOffset* value 0×84. This value can be converted to the UTC offset used since the timezone enabled bit is set. We do not count the timezone enabled bit, and get the value 0×04, meaning UTC+1:00 because each units corresponds to 15 min intervals.

For more information about the exFAT file system please see Schullich (2009).

1.2. Research problem

Since the exFAT file system is supported on all main desktop operating systems (Bretel, 2017), can we be sure that the specifications are followed by the exFAT file system developers for each of these platforms? Previous research has not tested exFAT implementations on multiple platforms, and our contribution will bridge that gap. A new feature of the exFAT compared to the FAT32 is the *UTCOffset* field for each timestamp. Since the specifications describe that the value stored is the offset from UTC to local time including

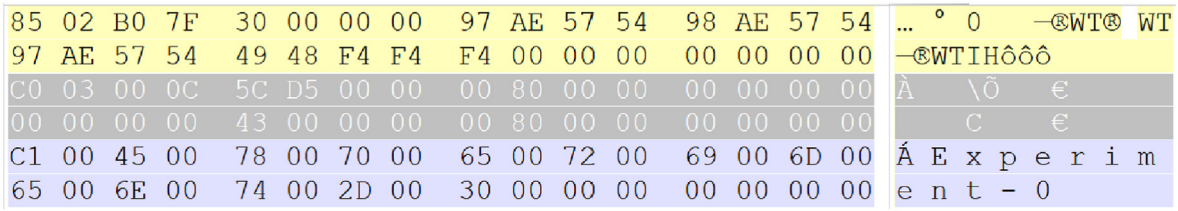

Fig. 1. ExFat set of directory entries.

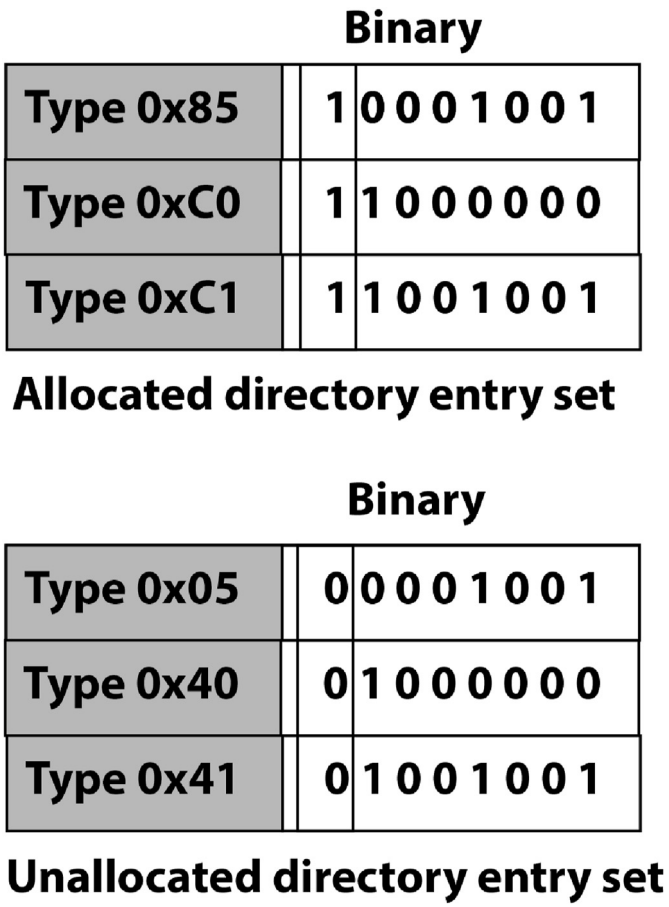

Fig. 2. ExFat allocation of directory entries.

any daylight adjustments Microsoft (2021b), it is relevant in an investigative context to test if we can find the timezone of the

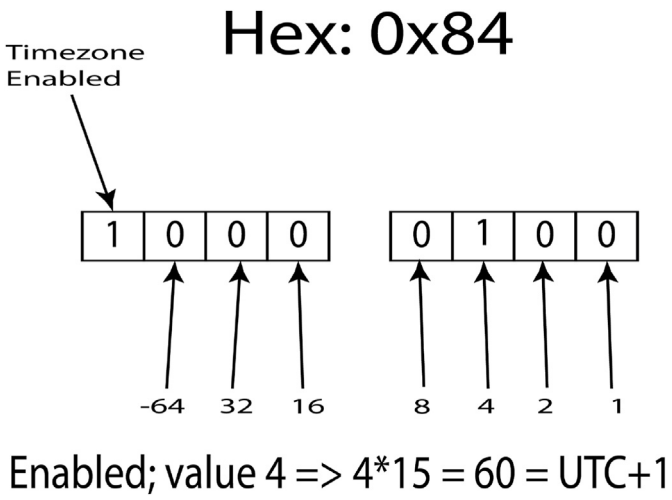

Fig. 3. ExFat timeszone field, from hex byte to UTC offset.

computer used to store the data on the file system by interpreting the file system metadata only. Currently, most Digital forensic tools claim support for the exFAT file system, but is this support accurate and reliable, and can it be validated for law enforcement purposes? Do the latest versions of the tools follow the exFAT specifications, and how do they handle exFAT implementations that do not comply with the exFAT specifications? The problems described above are defined as the following research questions:

- How do current exFAT implementations store timestamps?
- Can we use the UTC offset stored in a directory entry to describe the local time of the computer?
- Do current forensic tools interpret exFAT timestamps differently?

The main hypothesis:

Table 1
File directory entry (all fields are mandatory).

Field Name	Offset	Size	Value example	Comments
EntryType	0×0	0×1	0×85	Regular primary directory entry in use
SecondaryCount	0×01	0×01	0×04	Number of secondary directory entries in this set
SetChecksum	0×02	0×02		A checksum of all bytes (except this field) of directory entry set
FileAttributes	0×04	0×02	0×20	DOS mode: Archive and a file
Reserved1	0×06	0×02	0×00	Reserved
CreateTimestamp	0×08	0×04	0xAA493A54	Created 2022-01-26 09:13:20
LastModifiedTimestamp	0×0C	0×04	0xAA493A54	Modified 2022-01-26 09:13:20
LastAccessedTimestamp	0×10	0×04	0xAA493A54	Accessed 2022-01-26 09:13:20
Create10msIncrement	0×14	0×01		10 ms increments for Create
LastModified10msIncrement	0×15	0×01		10 ms increments for Modified
CreateUtcOffset	0×16	0×01	0xF4	Valid, UTC-3
LastModifiedUtcOffset	0×17	0×01	0xF4	Valid, UTC-3
LastAccessedUtcOffset	0×18	0×01	0xF4	Valid, UTC-3
Reserved2	0×19	0×07	0×00	Reserved

- H_1: the local time (the base truth) is related to the time stored within the primary directory entry when the timezone offset is valid.

The null hypothesis:

- H_0: the local time is not related to the timezone offset in the primary directory entry when the timezone offset is valid.

The $\alpha = 0.01$ is the confidence level, meaning if less than 1 percent of the observations supports the null hypothesis, then the null hypothesis is falsified, giving additional strength to the corresponding main hypothesis.

We have only defined one main hypothesis that is related to all three research questions, and the main hypothesis is based on an assumption that the exFAT specifications are followed, both by exFAT driver developers and by digital forensic tool developers.

1.3. Organisation of this paper

In the Introduction section we have introduced the importance of interpreting file system metadata, especially timestamps, in an investigative context, and we have given a short introduction to the exFAT file system. In addition, we have defined the research problems. In the Related Work section we summarise the current state of the art research related to exFAT. In the Methodology section we describe the methods we used for the experiments for all the supported operating systems (Windows, MacOS, and Linux), and in the Results section we describe our results. Then we discuss our results in the Discussion section, and we conclude in the Conclusion and Further Work section.

2. Related Work

Hamm (2009) was of the first to publish documentation about the exFAT file system structures. He described the file system from the view of a Digital Forensic practitioner. The exFAT system is similar to the old FAT systems, but each file has a set of directory entries which describe metadata of files. The file allocation table is mainly used for fragmented files, and an allocation bitmap file for describing which cluster (block) is allocated. Hamm (2009) describes that exFAT was first introduced in Windows CE in 2006, and then in Vista SP1 in 2008. In 2009 Windows XP drivers for exFAT were released. This work was performed before Microsoft released the full specifications, and the work was based on the patent US 20090164440 A1 (Microsoft, 2009), which describes most of the file system structures and their meaning.

Schullich (2009) continued the work from Hamm (2009) and described reverse engineering of the exFAT file system utilising black box analysis, using existing documentation such as patents, examination of other file systems in the FAT family, Google searches, Microsoft knowledge base, and low level examination of the exFAT file system. Schullich (2009) also developed a C program to output metadata structures. Files were created, added, deleted, and added again to observe the effect these operations had on the file system. The output of the C program was compared to the output of native Windows program such as *dir*, *chkdsk*, disk management, and *Windows Explorer*. Schullich (2009) also describes the internals of exFAT and its metadata structures. The timezone value fields (*UTCOffset*) were found based on experiments and observations, they were not described in the patents. These fields describe the timezone offset in units of 15 min.

Munegowda et al. (2012) describe allocation strategies for exFAT and compares them to FAT32. The exFAT file system uses the allocation bitmap (they call it the cluster heap) to search for free clusters. If enough free contiguous clusters are available for allocation, then the "No FAT Chain" is set to 0, the allocation bitmap is set for the new allocated clusters, and the FAT is not used. The stream extension directory entry points to the first cluster. However, if it is not possible to allocate contiguous clusters the "No FAT Chain" is set to 1, the allocation bitmap is updated with the new allocated clusters, and the FAT is used.

Unfortunately, the use of the "No FAT Chain" is misinterpreted by Munegowda et al. (2012), since the specifications from Microsoft (2021b) state that the *NoFatChain* field is set to 1 if the clusters are contiguous, and 0 if the FAT cluster chain is in use.

Munegowda et al. (2014) describe how exFAT can implement directory compaction techniques when its first cluster only has deleted/unallocated entries. In this case the directory entry for this directory should be changed to point to the next cluster, and the previous first cluster should be marked free in the allocation bitmap and in the file allocation table (compaction).

Ma et al. (2015) describe different approaches for data recovery for the exFAT file system. An unallocated file will change the set of directory entries from the types 0×85, 0xC0, and 0xC1, to the types 0×05, 0×40, and 0×41. One approach is using the second directory entry (stream extension directory entry, here type 0×40) of an unallocated file. Read its cluster start, find the start sector and extract the size of the file. If the file is not stored in contiguous clusters and not in the FAT (if damaged), then the file may not be completely recovered. Ma et al. (2015) also describe carving using signatures, and a machine learning approach utilising a Support-Vector Machine (SVM) classification algorithm.

Vandermeer et al. (2018) describe how a set of exFAT directory entries can be unallocated not necessarily as a result of deletion. By combining information from the allocation bitmap it is possible to differentiate between renamed, moved or deleted files. All these scenarios will have a set of unallocated directory entry sets, but only the deletion scenario will also set the bits in the bitmap for the corresponding clusters to zero. If the clusters of an unallocated file is allocated in the bitmap, the file may just as well be moved or renamed. If the file clusters are zeroed out in the allocation bitmap, then the file is deleted. They also proposed a methodology for recovering deleted files.

Heeger et al. (2021) describe anti-forensic techniques to hide data in the exFAT file system. They suggest the hiding of encrypted data in the *Create10msIncrement* and *LastModified10* ms-*Increment* fields, by only using six of the least significant bits in these single byte fields. In addition, one of the two most significant bits are set or none of these two bits are set. The *SetChecksum* field in the file directory entry is updated to take the hidden data into consideration. This process is done for all necessary directory entry sets used. Heeger et al. (2021) also suggest another approach called exHide which only uses metadata from deleted files. This approach uses the *Create10msIncrement* field (6 bits), and 1 bit from *CreateTimestamp* and *LastModifiedTimestamp* are used in order to create one byte. In addition, the FirstCluster and file size fields ValidDataLength and DataLength are used (the two latter needs to be equal). A total of 4 bytes are used for hiding for each metadata structure. For the exHide approach the *LastModified10msIncrement* was not used because Windows does not use this field when writing to the File System (Heeger et al., 2021).

3. Methodology

We used Linux Ubuntu 20.04 (using exFAT fuse v. 1.3), and Ubuntu 20.04 (using the native kernel exFAT driver), MacOS Monterey and Windows 10 as target operating systems (OSes).

We repeated the Linux experiments after removing the exFAT fuse package in Linux Ubuntu 20.04 to enforce the usage of the

native kernel exFAT driver.

All experiments are described in this section and illustrated in Fig. 4. The experiments A, C, and D were performed using a bash shell script in MacOS and Linux, and a batch script was used in Windows 10. Experiment B, and E were manually performed to simulate normal user activity. We have shared the scripts and the forensic images (Nordvik, 2022).

An overview of all experiments are shown in Fig. 4.

3.1. Experiment A - base

First we wiped the storage device, then we formatted it using the exFAT v. 1.0 file system. We performed experiments by utilising four different timezones, and for each timezone 100 files were created in their own directory on the USB storage device. Before each timezone change we performed an unmount and mount to make sure the data was written to the device. The local time as seen by the user and a timezone index (from 0 to 3) was encoded as a part of the filenames. After the experiments were performed a forensic image was created for each target OS/driver. Metadata for the created files were observed and timestamp related information was interpreted.

The timezones index includes:

- Europe/Moscow (index 0, UTC+3). Files stored in the directory Experiment-0.
- America/Godthab (index 1, UTC-3). Files stored in the directory Experiment-1.
- Atlantic/Azores (index 2, UTC-1). Files stored in the directory Experiment-2.
- Europe/Oslo (index 3, UTC+1). Files stored in the directory Experiment-3.

For the Windows OS we used similar timezone values;

- Russian Standard Time (index 0, UTC+3). Files stored in the directory Experiment-0.
- E. South America Standard Time (index 1, UTC-3). Files stored in the directory Experiment-1.
- Azores Standard Time (index 2, UTC-1). Files stored in the directory Experiment-2.
- W. Europe Standard Time (index 3, UTC+1). Files stored in the directory Experiment-3.

The following forensic images were created in this experiment:

```
ExFAT-Base-Experiment-A-Linux-
    ↪ NativeExfat.E01
ExFAT-Base-Experiment-A-Linux.E01
ExFAT-Base-Experiment-A-MacOS.E01
ExFAT-Base-Experiment-A-Windows10.E01
```

3.2. Experiment B - mounting and unmounting only

We used the Linux base forensic image from experiment A, restored to the USB storage device using *ewfmount* and *dd* commands, then the timezone was changed to Europe/Oslo (UTC+1) for MacOS, and W. Europe Standard Time (UTC+1) for Windows, which is different from each base experiment 0, 1, 2. The storage device was mounted on MacOS or Windows 10. We also restored the Windows base forensic image, and mounted and unmounted the USB storage device on Linux.

We did not make any change to any file. Then we unmounted the device and created a forensic image for each target OS/driver.

The following forensic images were created in this experiment:

```
ExFAT-Experiment-B-Linux-MountedLinux-
    ↪ NativeExfat.E01
ExFAT-Experiment-B-Linux-MountedMacOS.
    ↪ E01
ExFAT-Experiment-B-Linux-MountedWindows.
    ↪ E01
ExFAT-Experiment-B-MacOS-MountedWindows.
    ↪ E01
ExFAT-Experiment-B-Windows-MountedLinux.
    ↪ E01
```

3.3. Experiment C - accessing selected files

We selected the Linux base image when targeting MacOS, Windows and Linux native exFAT drivers, and we selected the Windows base image when targeting Linux exFat fuse driver. We restored the base forensic images to the USB storage device using *ewfmount* and *dd* commands, and we re-mounted it on one of the other operating systems after changing the timezone to America/New_York (UTC-5). We opened the files in *TextEdit* on MacOS, *Notepad* in Windows 10, and *Gedit* in Linux and then closed each file. Then we created a forensic image for each target OS/driver. We did not change or save any content.

The following forensic images were created in this experiment:

```
ExFAT-Experiment-C-Linux-LinuxOpenFiles-
    ↪ NativeExfat.E01
ExFAT-Experiment-C-Linux-MacOpenFiles.
    ↪ E01
ExFAT-Experiment-C-Windows-
    ↪ LinuxOpenFiles.E01
ExFAT-Experiement-C-Linux-
    ↪ WindowsOpenFiles.E01
```

3.4. Experiment D - changing the content of all files

We performed experiments to change the files to simulate normal user activity. We selected the Linux forensic image from the base experiments, restored to the USB storage device using *ewfmount* and *dd* commands, then re-mounted the device on one of the other operating systems. For every file in each of the 4 directories, we changed the content using the timezone America/

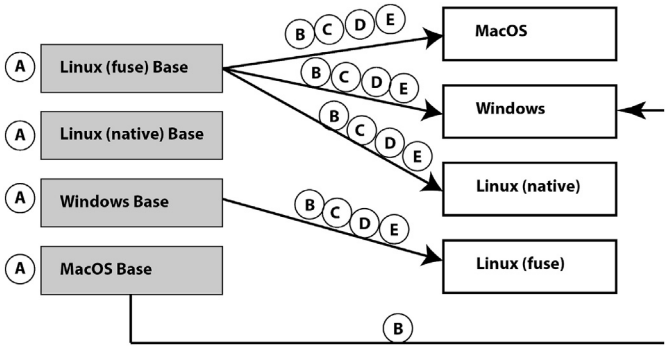

Fig. 4. Overview of all experiments, and all of them have a forensic image associated.

New_York (UTC-5) for Linux and MacOS, and Eastern Standard Time (UTC-5) for Windows which is different from each base experiment. Then we created a new forensic image for each target OS/driver. The metadata changes of the files were observed and documented.

The following forensic images were created in this experiment:

```
ExFAT-Experiment-D-Linux-Linux-Overwrite
    ↪ -Files-NativeExfat.E01
ExFAT-Experiment-D-Linux-Windows-
    ↪ Overwrite-Files.E01
ExFAT-Experiment-D-Windows-Linux-
    ↪ Overwrite-Files.E01
ExFATExperiment-D-Linux-MacOS-Overwite-
    ↪ Files.E01
```

3.5. Experiment E – changing the content of selected files

We selected the Linux base image, restored it to the USB storage device using *ewfmount* and *dd* commands, changed the timezone to America/New_York (UTC-5) for Linux and MacOS and Eastern Standard Time (UTC-5) for Windows, and re-mounted it on one of the other operating systems. We changed the content of selected files manually by opening them in TextEdit for MacOS, Notepad in Windows 10 (the timezone had entered daylight time, meaning UTC-4), or Gedit in Linux. Then we wrote a word and saved and closed each file. We only changed the files in the directory Experiment-0.

Then we created a forensic image for each target OS/driver. The following forensic images were created in this experiment:

```
ExFAT-Experiment-E-Linux-Linux-Overwrite
    ↪ -Files-Manually-NativeExfat.E01
ExFAT-Experiment-E-Linux-MacOS-Overwrite
    ↪ -Manually.E01
ExFAT-Experiment-E-Linux-Windows-
    ↪ Overwrite-Manually.E01
ExFAT-Experiment-E-Windows-Linux-
    ↪ Overwite-Manually.E01
```

3.6. Tool testing

If the tool supported changing timezone, we adjusted to the timezone stored in the hex dump for each experiment. We tested the following Digital Forensic (DF) tools:

- Autopsy v. 4.19.3 (Windows version)
- FTK Imager v. 4.5.0.3
- X-Ways Forensics v. 20.04 SR-4
- EnCase Examiner v. 22.1

The FTK Imager does not support changing timezone, while Autopsy, X-Ways and EnCase do.

When testing the tools we did not only compare the result from different tools, but we also assessed the results manually in the directory entries of the exFAT file system. This manual verification was necessary since dual tool verification is not a reliable method (Nordvik et al., 2021; Knight and Leveson, 1986).

3.7. Limitations and assumptions

We assumed that the current implementations of exFAT v. 1.0 store timestamps as localtime, and that each *UTCOffset* field describes the deviation between the local time and the UTC, which the exFAT specifications describes (Microsoft, 2021b). However, an implementation may choose not to utilise the *UTCOffset* fields.

We did not consider file systems that are manipulated, attacked, or where anti-forensic methods as described by Wani et al. (2020) are used. The experiment methodology described includes the actions that were performed on the USB storage device.

We also assume the exFAT file system interpretation by forensic tools was unreliable until we have verified the findings (Zero Trust (Neale et al., 2022)).

4. Results

4.1. Experiment A - creating files on an exFAT storage

We observed, as shown in Table 2, that for MacOS the timestamps were stored on disk in UTC-3 when the local time was UTC+3 (Europe/Moscow), and similar were the UTC offset switched from negative to positive for timezones with negative UTC offset.

If we normalise the stored timestamps to UTC-0, we can see that all files are created in the period 24/02/2022 00:52 to 00:53. The files were created using a script, explaining why they were near in creation time.

It was interesting to observe that the latest timezone used on MacOS also changed the last access time for all files, even for files not accessed.

Table 3 shows that the Windows exFAT driver is following the exFAT specifications when setting the *UTCOffset* fields. In this case the local time (real time) was stored.

Table 4 shows the results for the Linux Ubuntu experiment with the exFAT fuse driver, and it set the *UTCOffset* fields to 0×00, meaning these fields are not valid because the most significant bit is not set. We observed that the timestamps are stored using localtime. However, it is not possible to interpret what the local time UTC offset was by only assessing the stored timestamps and the *UTCOffset* fields.

Table 5 shows the results for the Linux Ubuntu experiment using the native exFAT driver, and it sets the *UTCOffset* fields to 0×80, meaning these fields are valid and is set to UTC+0. The experiment shows that we cannot interpret what the local time UTC offset was by assessing only the stored timestamps and the *UTCOffset* fields.

The Experiment A shows that:

- 400 of 1600 observations show the usage of the 0×00 invalid *UTCOffset* value. Invalid values are excluded from hypothesis testing.
- 400 of 1200 (33 percent) show the usage of 0×80 valid value, even when local time deviates from UTC+0.
- 800 of 1200 (67 percent) observations take the local time into consideration when storing timestamps and UTC offsets.

This means that our main hypothesis is not true for all operating systems.

4.2. Experiment B - mounting exFAT storage

The result shown in Table 6 shows that MacOS will change the *UTCOffset* of the last accessed timestamp for all files when a USB storage device is mounted and unmounted. It also shows that when

Table 2
Experiment A Results - MacOS. We can see that stored timestamps use a timezone offset with switched signs compared to the computer the experiments were executed on.

Base TZ	Action	Stored TZ	Stored Time	Real TZ	Real Time	Observations
Europe/Moscow	Created	0xF4 (UTC-3)	23/02/2022 21:52	UTC+3	24/02/2022 03:52	100
America/Godthab	Created	0×8C (UTC+3)	24/02/2022 03:53	UTC-3	23/02/2022 21:53	100
Atlantic/Azores	Created	0×84 (UTC+1)	24/02/2022 01:53	UTC-1	23/02/2022 23:53	100
Europe/Oslo	Created	0xFC (UTC-1)	23/02/2022 23:53	UTC+1	24/02/2022 01:53	100

Table 3
Experiment A Results - Win10. We can see that all types of timestamps are stored using the local UTC offset of the computer the experiments were executed on, and that real time is the same as stored time.

Base TZ	Action	Stored TZ	Stored Time	Real TZ	Real Time	Observations
Russian Standard Time	Created	0×8C (UTC+3)	24/02/2022 21:23	UTC+3	24/02/2022 21:23	100
E. South America Standard Time	Created	0xF4 (UTC-3)	24/02/2022 15:23	UTC-3	24/02/2022 15:23	100
Azores Standard Time	Created	0xFC (UTC-1)	24/02/2022 17:24	UTC-1	24/02/2022 17:24	100
W. Europe Standard Time	Created	0×84 (UTC+1)	24/02/2022 19:24	UTC+1	24/02/2022 19:24	100

mounted on MacOS the directories *.fseventsd* and *.SpotLight-V100* were created. When mounted on Windows, the directory *System Volume Information* directory was created.

4.3. Experiment C - opening files

In the Experiment C in Table 7, we used the timezone America/New_York (UTC-5). The last accessed timestamp was changed and stored using UTC-5 (local time), but the *UTCOffset* fields were not changed in Linux exFAT fuse driver. Since we used the Windows base forensic image, and *UTCOffset* fields were not touched in Linux (exFAT fuse) when opening files using *Gedit*, the local timestamp stored does correspond with the preserved *UTCOffset* fields for the last modified and the created, but not necessarily the last accessed. We illustrate this in Fig. 5.

For the Linux native driver we used the base of the Linux exFAT fuse, and here all *UTCOffset* fields were changed to 0×80 when using the Linux exFAT native driver, even though only the timestamp last accessed was changed. This change shows that the native driver interprets the previous value 0×00 *UTCOffsets* fields as timestamps stored in UTC+0, which was an incorrect assumption for all our experiments. We illustrate this in Fig. 6.

It was strange that the last modified timestamp was changed when files were opened using TextEdit on MacOS, especially because we did not change the content of any files. The timestamp for last modified was just converted to use the local UTC offset of the computer. The previous last modified timestamp was assumed to be UTC-5 (our local UTC offset) and then the timestamp was converted to UTC+5. This must fail since the Linux base did not use *UTCOffset* fields, and the UTC-5 assumption was wrong. We illustrate this in Fig. 7. In Windows no change at all was registered when just opening files in Notepad. We illustrate this in Fig. 8.

4.4. Experiment D and E: changing exFAT files on multiple OSes

The results are shown in Table 8. We can see that Windows set the *LastModified10msIncrement* to 0×00 when changing the files in Windows, and the last modified and last access timestamps and the corresponding *UTCOffset* fields are updated.

Linux (exFAT fuse driver) only changes the last modified timestamp and the *LastModified10msIncrement* (values 0×00 or 0×64), but not the *UTCOffset* fields when changing the files using the bash script for appending more text in each file. The last accessed timestamp was not changed. However, when using Gedit to change the file content in Linux, it set all timestamps to the change time using the local time of the computer and sets all *UTCOffset* fields to 0×00. The *10msIncrement* fields are also updated. Since all timestamps are equal, it looks like the file was created at the time it actually was only changed. The observations are similar when using the Linux Ubuntu native exFAT driver, except that the *UTCOffset* fields are set to 0×80, and that not only 0×64 and 0×00 are used for the *10msIncrement* fields. For both Linux drivers, the original create timestamp is lost.

MacOS also behaves differently if the content is changed using piping in a bash script, or if the files are changed by manually opening and changing the files in TextEdit. The latter will even try to set the *UTCOffset* for the created timestamp, assuming the original timestamp was stored with the same timezone offset as the local computer (in our case UTC-5), which was an incorrect assumption in our case. We also observed that additional fork files were created when changing the files in TextEdit, but not when using the bash script. These fork files are pre-pended with and may contain metadata information that describes which app was used to change the file. A fork is a named attribute used normally in HFS or APFS that contains a stream of data, and is similar to alternate data streams in NTFS (Wani et al., 2020).

4.5. 10 msIncrement fields

Another result we observed was the usage of the 10 ms granularity fields; *Create10msIncrement* and *LastModified10msIncrement*. Based on the results in Table 9 we can verify that Windows 10 does

Table 4
Experiment A Results - Linux Ubuntu 20.04 using exFAT fuse v.1.3. We can see that the timestamps are stored using the local time of the computer the experiments were executed on, and that real time is the same as stored time. However, the *UTCOffset* fields are not in use.

Base TZ	Action	Stored TZ	Stored Time	Real TZ	Real Time	Observations
Europe/Moscow	Created	0×00 (Not used)	02/03/2022 16:11	UTC+3	02/03/2022 16:11	100
America/Godthab	Created	0×00 (Not used)	02/03/2022 10:12	UTC-3	02/03/2022 10:12	100
Atlantic/Azores	Created	0×00 (Not used)	02/03/2022 12:12	UTC-1	02/03/2022 12:12	100
Europe/Oslo	Created	0×00 (Not used)	02/03/2022 14:13	UTC+1	02/03/2022 14:13	100

Table 5
Experiment A Results - Linux Ubuntu 20.04 using exFAT native driver. We can see that the timestamps are stored using UTC+0, not the local time of the computer the experiments were executed on. The *UTCOffset* fields are used (set to 0 × 80).

Base TZ	Action	Stored TZ	Stored Time	Real TZ	Real Time	Observations
Europe/Moscow	Created	0×80	16/03/2022 14:48	UTC+3	16/03/2022 17:48	100
America/Godthab	Created	0×80	16/03/2022 14:49	UTC-3	16/03/2022 11:49	100
Atlantic/Azores	Created	0×80	16/03/2022 14:49	UTC-1	16/03/2022 13:49	100
Europe/Oslo	Created	0×80	16/03/2022 14:50	UTC+1	16/03/2022 15:50	100

Table 6
Experiment B Results - MacOS. Impact of mounting and unmounting

OS	Action	Timestamp	10msIncrement	UtcOffset	Observations	New Directories
MacOS	Mount/unmount	LA	Not changed	LA (switched sign)	400	.fseventsd,.SpotLight-V100
Windows	Mount/unmount	Not changed	Not changed	Not changed	400	System Volume Information
Linux	Mount/unmount	Not changed	Not changed	Not changed	400	

Table 7
Experiment C Results. Impact of opening files.

OS	Action	Tool used	Timestamp	10msIncrement	UtcOffset	Observations
Linux (fuse)	Open	Gedit	LA using local time	Not changed	Not changed	400
Linux (native)	Open	Gedit	LA using UTC+0	Not changed	All to 0×80	400
MacOS	Open	TextEdit	LM(*) and LA using local time	Not changed	LM and LA (switched)	400
Windows	Open	Notepad	Not changed	Not changed	Not changed	400

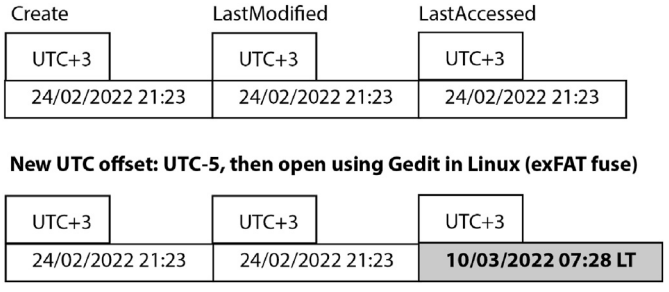

Fig. 5. Changes in timestamps and *UTCOffset* fields when opening a file using Gedit in Linux Ubuntu 20.04 exFAT fuse driver. The *LastAccessedTimestamp* is changed using local time (LT), which was UTC-5, however the *LastAccessedUtcOffset* is not changed. In this case the last accessed is inaccurate.

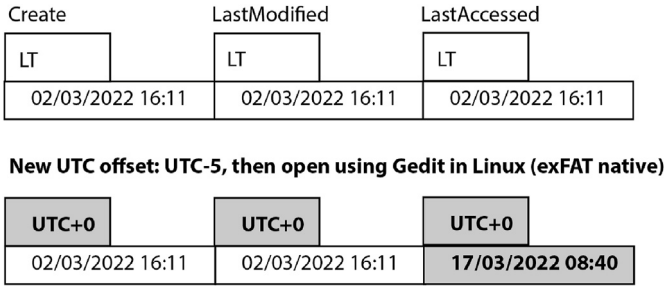

Fig. 6. Changes in timestamps and *UTCOffset* fields when opening a file using Gedit in Linux Ubuntu 20.04 exFAT native driver. The *LastAccessedTimestamp* is stored as UTC+0, however the *CreateUtcOffset* and *LastModifiedUtcOffset* are also changed to UTC+0, but not the timestamps. In this case the create and last modified timestamps are inaccurate.

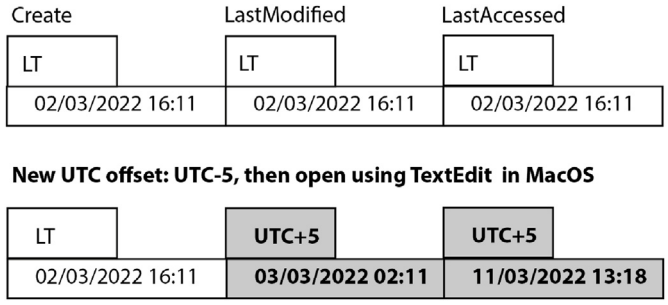

Fig. 7. Changes in timestamps and *UTCOffset* fields when opening a file using TextEdit in MacOS Monterey. The *LastAccessedTimestamp* is stored as UTC+5, even though the real timezone was UTC-5. The *CreateUtcOffset* is not changed, but the *LastModifiedUtcOffset* is changed to UTC+5, trying to convert LT from UTC-5 to UTC+5. In this case the last modified timestamp is inaccurate.

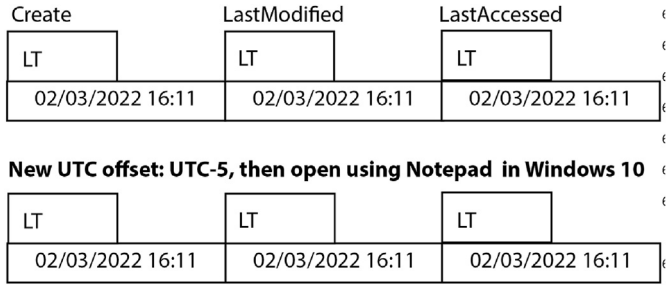

Fig. 8. Changes in timestamps and *UTCOffset* fields when opening a file using Notepad in Windows 10. Nothing was changed. In this case the *LastAccessedTimestamp* is inaccurate.

not update the *LastModified10msIncrement* accurately on change, as described by Heeger et al. (2021). Windows 10 sets it to 0x00. However, MacOS does update these fields. We also observed that Linux update these fields, and we observed that both *10msIncrement* fields either had the value 0×00 or 0×64 for the exFAT fuse driver. The latter value 0×64 is 100 in decimal, meaning in this context 1000 ms or 1 s. However, the native exFAT driver used by Ubuntu 20.04 updates the *10msIncrement* fields similar to MacOS.

Table 8
Experiment D and E Results - Changes in Timestamps, 10msIncrement and *UTCOffset* fields in the exFAT file directory entry when changing the files on Windows, MacOS or Linux.

OS	Action	Tool used	Timestamp	10msIncrement	UtcOffset	Observations
Windows	Change	≫	LM and LA	LM (0×00)	LM and LA	400
Windows	Change	Notepad (manual)	LM and LA	LM (0×00)	LM and LA	100
MacOS	Change	≫	LM and LA	LM	LM and LA	400
MacOS	Change	TextEdit (manual)	C, LM and LA	LM	C, LM and LA	100
Linux (fuse)	Change	≫	LM	LM (0×00 or 0×64)	Not changed	400
Linux (native)	Change	≫	LM	LM	All is set to 0×80	400
Linux (fuse)	Change	Gedit (manual)	C, LM, and LA	C and LM (0×00 or 0×64)	All is set to 0×00	100
Linux (native)	Change	Gedit (manual)	C, LM, and LA	C and LM	All is set to 0×80	100

4.6. Tool testing

Law enforcement require tools that give accurate results, and interpret timestamps correctly, else any incorrect results may impact a criminal case. Therefore, we will show how different Digital Forensic tools show timestamps from the exFAT file system, in the context of the above mentioned experiments.

4.7. Autopsy

When testing Autopsy we adjusted the timezone for each experiment in order to match the timezone used for storing the timestamp. The date/time should match between stored time and the time shown in Autopsy, and the results are shown in Table 10. We can see that experiment index 2 matches where both timezones are using UTC+1. We also need to take into consideration that Autopsy was initially set to the timezone Europe/Oslo (UTC+1) in standard time when adding the forensic image. Therefore, we found that Autopsy interprets the exFAT timestamps using this initial timezone as the stored local time. In the other experiments we saw that Autopsy interpreted all stored timestamps as the initial local time (here UTC+1), and then tries to convert it to a timezone selected in Autopsy options view tab. This assumption about the current timezone was incorrect in most of our experiments. For instance, in order to change to UTC-3, Autopsy tries to subtract −4 from the stored timestamp, since it assumes the stored timestamp is given in UTC+1. For UTC+3 Autopsy adds 2 h to get from UTC+1 to UTC+3. The only place it gives the same timestamp is when the UTC offset is the same as the assumed timezone. However, even in the latter case it is inaccurate, because Atlantic/Azores is using UTC-1 in standard time as seen in Table 10.

When we added the Windows exFAT forensic image to Autopsy, we adjusted the initial value to Europe/Moscow (UTC+3). Autopsy tried to adjust the timezone based on the initial UTC+3 that it interpreted as the local time stored. This assumption is only correct for the Experiment-0 files.

4.8. FTK imager

In Table 11 we were not able to adjust the timezone shown by FTK Imager. Instead, the tool converted the timestamps to UTC+0. The conversions from stored timestamp to UTC+0 was correct. However, when adding the linux base image from the exFAT fuse driver with *UTCOffset* fields not valid, then the Created, Modified and Accessed are set to N/A (Not Applicable). The latter approach is fine, since it is infeasible to show the date and time when the UTC offset fields are not valid. However, showing the timestamps with a LT (Local Time) would have been better.

4.9. X-Ways Forensics

Table 12 shows the timestamps correctly using the same UTC offset as they were stored. X-Ways also displays the applied UTC offset after each timestamp, as shown in Fig. 9 and the hex representation of the first file in Fig. 10. X-Ways converts the exFAT timestamp correctly using any selected timezone. When X-Ways interprets *UTCOffset* fields with a mix of valid and invalid values, it tries to convert all values using the stored UTC offset to compute the selected timezone UTC offset. However, converting an invalid UTC offset value to a timezone UTC offset is based on an assumption about the previous stored UTC offset.

4.10. EnCase forensic

EnCase shows the timestamps in the selected timezone taking the *UTCOffset* fields into consideration, as shown in Table 13. When it comes to interpreting an exFAT filesystem created by the Linux Ubuntu exFAT fuse driver, EnCase interprets the stored timestamp as UTC+0 and tries to convert to the selected timezone, even though the *UTCOffset* fields have the 0×00 value (not valid). This is only accurate if the local time of the Linux computer was UTC+0, which it was not in all our experiments.

When there were mixed values in the *UTCOffset* fields, EnCase correctly showed timestamps where *UTCOffset* fields contained valid values, but failed if these values were invalid (0 × 00). For instance, Encase correctly showed the ones with value 0×80 using the selected timezone in EnCase, however the 0×00 value was wrongly interpreted as if the timestamps are stored using UTC+0.

Based on the experiments we can see that EnCase can be validated for exFAT as long as the *UTCOffset* fields are valid. If they are not valid, then EnCase seems to make an assumption about the UTC offset that may be wrong.

We can see this interpretation in Fig. 11 where the local time stored was 02.03.2022 at 16:11 (UTC+3), but wrongly interpreted

Table 9
Experiment Results - Usage of the 10 ms granularity fields in the exFAT file directory entry when using Windows, MacOS or Linux.

OS	Create10msIncrement	LastModified10msIncrement	Observations
Windows	In use	Not used, set to 0x00	400
Mac OS	In use	In Use	400
Linux	In use	In Use	400

Table 10
Experiment Results - MacOS and Autopsy v. 4.19.3

Base TZ (Index)	Type	Stored TZ	Stored Time	Autopsy TZ	Autopsy Time	Observations
Europe/Moscow (0)	Created	0xF4 (UTC-3)	23/02/2022 21:52	UTC-3	23/02/2022 17:52	100
America/Godthab (1)	Created	0×8C (UTC+3)	24/02/2022 03:53	UTC+3	24/02/2022 05:53	100
Atlantic/Azores (2)	Created	0×84 (UTC+1)	24/02/2022 01:53	UTC+1	24/02/2022 01:53	100
Europe/Oslo (3)	Created	0xFC (UTC-1)	23/02/2022 23:53	UTC-1	23-02-2022 21:53	100

Table 11
Experiment Results - MacOS and FTK Imager v. 4.5.0.3

Base TZ (Index)	Type	Stored TZ	Stored Time	FTK TZ	FTK Time	Observations
Europe/Moscow (0)	Created	0xF4 (UTC-3)	23/02/2022 21:52	UTC+0	24/02/2022 00:52	100
America/Godthab (1)	Created	0×8C (UTC+3)	24/02/2022 03:53	UTC+0	24/02/2022 00:53	100
Atlantic/Azores (2)	Created	0×84 (UTC+1)	24/02/2022 01:53	UTC+0	24/02/2022 00:53	100
Europe/Oslo (3)	Created	0xFC (UTC-1)	23/02/2022 23:53	UTC+0	23/02/2022 00:53	100

Table 12
Experiment Results - MacOS and X-Ways Forensics v. 20.04 SR-4.

Base TZ (Index)	Type	Stored TZ	Stored Time	X-Ways TZ	X-Ways Time	Observations
Europe/Moscow (0)	Created	0xF4 (UTC-3)	23/02/2022 21:52	UTC-3	23/02/2022 21:52	100
America/Godthab (1)	Created	0×8C (UTC+3)	24/02/2022 03:53	UTC+3	24/02/2022 03:53	100
Atlantic/Azores (2)	Created	0×84 (UTC+1)	24/02/2022 01:53	UTC+1	24/02/2022 01:53	100
Europe/Oslo (3)	Created	0xFC (UTC-1)	23/02/2022 23:53	UTC-1	23/02/2022 23:53	100

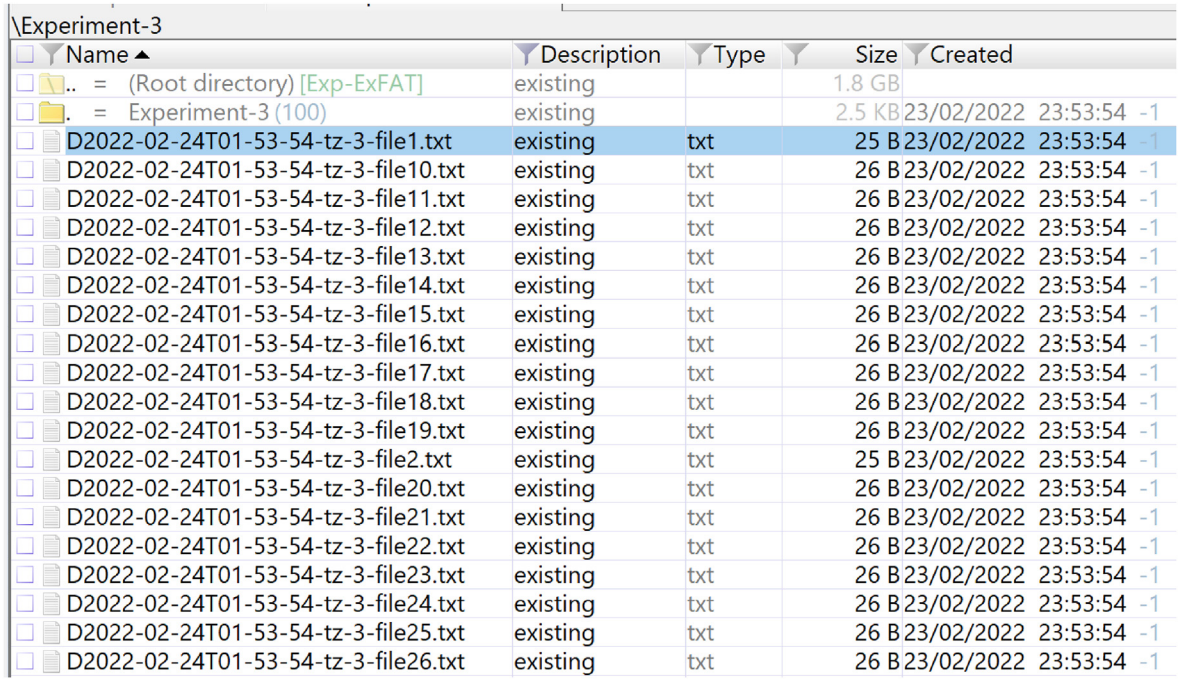

Fig. 9. ExFat timezones using stored UTC-1 offset for Experiment3 on MacOS and the X-Ways directory listing.

as UTC+0 because of the 0×00 values in the *UTCOffset* fields, and then EnCase adds 3 h to convert to UTC+3, which is incorrect in this context.

5. Discussion

The observations show that exFAT on Windows uses the local timezone offset including any daylight settings without switching signs, accurately storing the timestamp using the UTC offset of the local time of the computer. The MacOS experiments show that timestamps are not stored in the local time, instead it converts the UTC offset by switching signs. The Linux experiments using exFAT fuse driver shows that the timezone *UTCOffset* fields are not in use, and that the timestamp is stored using localtime, while the Linux native driver and *UTCOffset* fields are set to 0×80 (UTC+0) and the timestamps are stored using UTC+0 for the native exFAT driver.

The implementation used by MacOS will not make timestamps inaccurate when showing the files from these experiments in Windows. Windows File Explorer interprets the exFAT file system correctly, meaning File Explorer will take the timezone *UTCOffset*

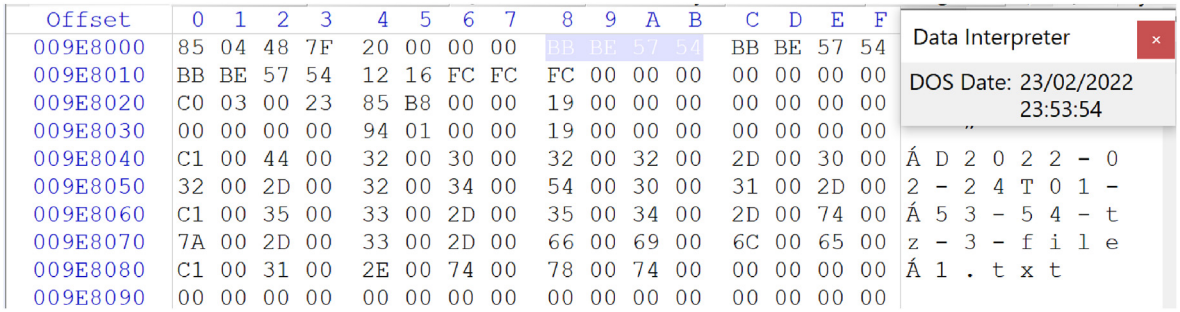

Fig. 10. ExFat timeszones using stored UTC-1 offset for Experiment3 and the X-Ways for D2022-02-24T01-53-54-tz-3-file1.txt.

Table 13
Experiment Results - MacOS and EnCase Forensic v. 22.1

Base TZ (Index)	Type	Stored TZ	Stored Time	X-Ways TZ	X-Ways Time	Observations
Europe/Moscow (0)	Created	0xF4 (UTC-3)	23/02/2022 21:52	UTC-3	23/02/2022 21:52	100
America/Godthab (1)	Created	0×8C (UTC+3)	24/02/2022 03:53	UTC+3	24/02/2022 03:53	100
Atlantic/Azores (2)	Created	0×84 (UTC+1)	24/02/2022 01:53	UTC+1	24/02/2022 01:53	100
Europe/Oslo (3)	Created	0xFC (UTC-1)	23/02/2022 23:53	UTC-1	23/02/2022 23:53	100

field into consideration before converting it to the local time used by the local computer. An example from Experiment-3 is shown in Fig. 12 for File Explorer. The same is true when mounting an exFAT storage device from Windows to MacOS, except that MacOS changes the Last Accessed timestamp and the *LastAccessedUtcOffset*. However, both MacOS and Windows take the *UTCOffset* fields into account and show them in the local time of the computer.

When it comes to Linux Ubuntu 20.04 exFAT fuse driver, no tools examined in this experiment will know what the timezone offset of the local time was for each file created. If the files are changed by using Gedit in Linux the *UTCOffset* are set to 0×00 and all timestamps are changed to the time of the update. Therefore, the original created date is lost. Linux Ubuntu 20.04 native exFAT driver store the time in UTC+0 no matter what the timezone was, and this is an implementation were the knowledge of the local time of the computer is not preserved. It is not necessarily an incorrect method, and it does not impact how the timestamps are presented in other OSes and in Digital Forensic tools. However, they are not following the specifications (Microsoft, 2021b).

Digital Forensic tools interpreting the timestamps should describe that the timestamps are stored as the localtime whenever the *UTCOffset* fields are not in use, and investigators cannot assume anything about which timezone was in use for a particular file when using the Linux exFAT fuse driver. If the *UTCOffset* fields was not in use, any change of timezone using a Digital Forensic tool should not change the time shown, but continue using the local time. However, if *UTCOffset* fields are valid, then Digital Forensic tools should change to the selected timezone utilising the necessary computation based on the stored timestamp and *UTCOffset* fields. Further, it should not be assumed that the *UTCOffset* fields only use one timezone offset.

If we know that an exFAT storage device has only been used on a MacOS, we can describe the local time of the computer (UTC offset for the timezone and any daylight settings) by switching the sign again. We can also find the last registered MacOS local computer UTC offset used by checking the *LastAccessedUtcOffset* field. On the other hand it will be difficult to assess if the timezone offset was initially set by a Windows or a MacOS computer. For instance, a MacOS using UTC-1 will store the timestamp in UTC+1, and a Windows computer using UTC+1 will store timestamps in UTC+1. In this scenario we have *UTCOffsets* of 0×84 on all timestamps,

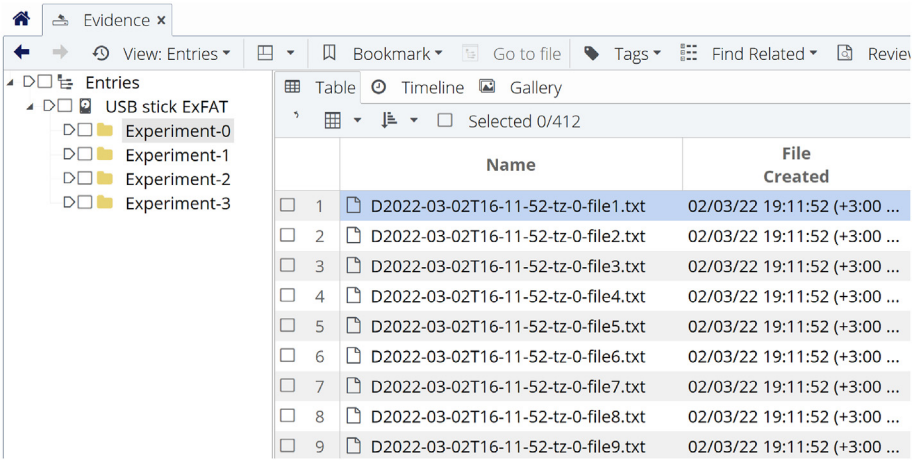

Fig. 11. ExFat timeszones using stored UTC+3 offset for Experiment0 from the Linux Base exFAT fuse image and the EnCase.

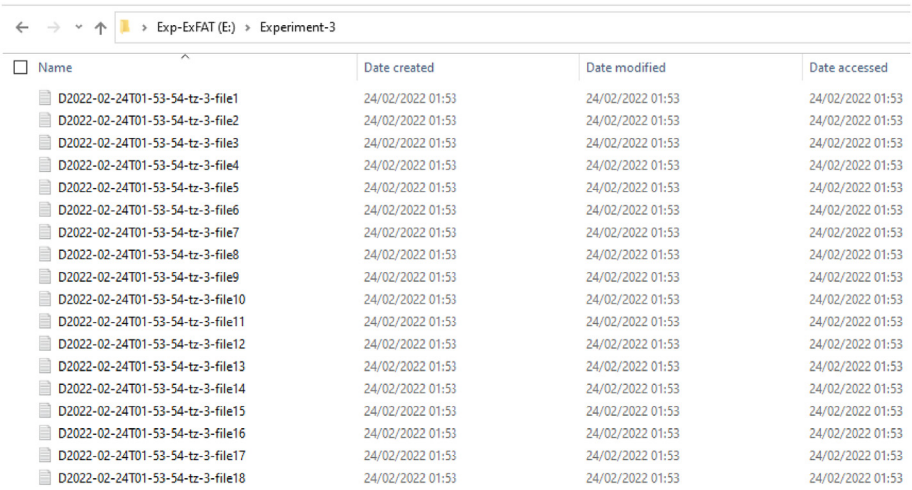

Fig. 12. ExFat timeszones using Europe/Oslo timezone (UTC+1) for Experiment-3 and the Windows 10 computer.

though they were actually running on two different timezones.

The existence of a *.fseventsd* and *.Spotlight-V100* directory within the root directory is an indication of MacOS usage, while existence of a *System Volume Information* directory within the root directory is an indication of Windows usage.

If only mounting and dismounting a USB storage device with exFAT file system on a MacOS, then the *LastAccessedUtcOffset* field will be updated and stored using the switching signs method on every file on the device. However, the Created and Modified timestamps and their corresponding *UTCOffset* fields will not be updated. Therefore, it is common that a storage device used on both on Windows and MacOS will include timezone offsets that deviate within the same directory entry, even when using the same timezone. It is also important to note how easy it is to change timestamps unintentionally by connecting a storage device to a MacOS without using a write blocker.

5.1. Rules for updating timestamps

Table 14 shows which operating system exFAT driver complies with the exFAT specifications for updating the different timestamps. The MacOS and Linux changes the timestamp for creation when using *TextEdit* or *Gedit* to change the content of a file. This means the timestamps are changed by the driver, and it may already be inaccurate before parsing and interpretations of Digital Forensic tools. Most of the drivers update the last modified on change, but MacOS may update the last modified on open only. The last accessed timestamp is updated on open for all exFAT drivers except on Windows when using *Notepad* to open files. In Linux when using a bash script to append content the last accessed is not updated for both exFAT drivers, only the last modified timestamp.

This unequal behaviour impact the investigation, and therefore it must be emphasized that it is important to understand which OS driver has been used in order to interpret the findings correctly.

5.2. 10 ms granularity

Even though the exFAT specifications describe that the *LastModified10msIncrement* field should be updated when updating any clusters used by the stream extension directory entry, or when changing the *ValidDataLength* or *DataLength* fields (Microsoft, 2021b), we observed that this was not implemented in Windows and implemented in MacOS.

The stego-only approach proposed by Heeger et al. (2021) will fail if the storage device is mounted and files are updated using MacOS or Linux, since this approach is using the LastModified10msIncrement. However, their exHide approach will work since they are not using this field and are utilising only unallocated directory entries.

5.3. Patterns

The three operating systems used in our experiments show distinct patterns which can be used to identify the OS used for a particular exFAT storage device. The most clear pattern is when the *UTCOffset* fields have the 0×00 value, meaning it is used by the Linux exFAT fuse driver. The other Linux exFAT fuse pattern is that the *10msIncrement* fields have the value 0×00 or 0×64. We should not see any System Volume Information directory or .fseventsd and .SpotLight-V100 directories if the USB storage is only used on Linux. We only know that the local time is used to store the timestamps when using the exFAT fuse driver, we do not know which timezone was used. The Linux exFAT native driver uses 0×80 (UTC+0) always, but other OSes using the GMT timezone will also use 0×80, and therefore this is not a good pattern to identify Linux.

MacOS uses all *UTCOffset* fields and both *10msIncrement* fields. In addition it creates the directories .fseventsd and .Spot-Light-V100. If no *UTCOffset* fields are 0×00 and the System Volume Information directory is not present in the root directory, then we know MacOS has been used. However, the native Linux driver may also have been used, but in this context if all files are using another timezone than GMT, then we know MacOS has been used. Another sign is the usage of fork files when using GUI apps to change files. The latter is very interesting since we can see which app was used to change the files. When only MacOS has been used, we can switch the stored *UTCOffset* sign to find the local time of the computer used when creating or updating a file.

Windows updates all *UTCOffset* fields and *Create10msIncrement*, and the *LastModified10msIncrement* is set to 0×00 on creation. In addition the directory System Volume Information is created in the root directory. On change the last modified and last accessed timestamps are updated, and the last modified is updated for the *10msIncrement* field, which is set to 0x00. If no *UTCOffset* fields have the value 0×00, and the directories .fseventsd and the .SpotLight-V100 are not present, and the System Volume Information directory is present, and all files have 0×00 for the *LastModified10msIncrement*, but uses the *Create10msIncrement*, then we know that Windows has been used. When Windows is the only OS

Table 14
Rules for updating timestamps - compliance.

Timestamp	Specs	Driver compliance
CreateTimestamp	On creation	Windows 10
LastModifiedTimestamp	Modifying cluster content	Windows 10, Linux
LastAccessedTimestamp	Modifying or reading cluster content	MacOS

used, then we can find the local time used by the computer by using the *UTCOffset* fields.

5.4. Challenges

The MacOS exFAT driver will try to update the Create timestamp when changing a file manually using TextEdit. It also made an assumption that the timezone must be equal the local time of the Mac computer. This may or may not be true, and if wrong will effectively change the created date to a wrong time.

Linux (both drivers) changed all timestamps when changing files using *Gedit*, using the local time when the change happened (3 equal timestamps). A text document created a year ago, will get a new set of equal timestamps for create, last modified, and last accessed when changing the file using *Gedit*. If the digital forensic investigator identify that the exFAT storage device has been used on Linux, we can not say anything about creation time.

5.5. Tools

In this section we discuss if different tools can be validated for law enforcement usage. With tool validation we mean if the tool is appropriate for its intended usage (ISO, 2017). Our aim is that the tool developers improve tools where we have found inaccuracy. This also means that in future releases of these tools, the interpretation may have been improved.

5.6. Autopsy

Autopsy interpreted that exFAT has stored the timestamp as the local time initially set when adding the forensic image into Autopsy, and does not consider the timezone offset in the directory entry. If the initial given local time does not match the stored local timezone *UTCOffset* for each timestamp, then it will yield erroneous results. Setting the initial local timezone correctly requires the DF investigator to verify the timezone *UTCOffset* manually in a hex viewer, but Autopsy cannot support files with multiple timezone offsets stored on the same file system.

Based on these findings we assess that Autopsy v. 4.19.3 (Windows version) cannot be validated for interpreting exFAT timestamps.

5.7. FTK-imager

FTK Imager displays the timestamps in UTC+0 by taking the *UTCOffset* fields into consideration. FTK Imager can be validated for interpreting exFAT timestamps as long as the timestamps shown are interpreted as UTC+0 by the digital forensic investigator. It is not suitable to use for a storage that have been using the Linux exFAT fuse driver, since it will not show any timestamps because of the non valid *UTCOffset* field values.

FTK Imager can be validated for interpreting exFAT timestamps.

5.8. X-ways

X-Ways displays the timestamps correctly in the timezone selected by the investigator, and it takes the stored *UTCOffset* fields into consideration when adjusting the time to the selected timezone. X-Ways also show the UTC offset used after each timestamp. X-Ways can be validated for interpreting exFAT timestamps as long as all *UTCOffset* fields are the same within the same directory entry.

If the *CreateUtcOffset* is 0×00 and the *LastModifiedOffset* and *LastAccessedUtcOffset* is a valid UTC offset, then it will try to show all timestamps in the selected timezone. However, it cannot know the local time of the timestamp using *UTCOffset* value 0×00, and any conversion must be based on assumptions. This is especially important when there are mixed *UTCOffset* values, where one or more contain the value 0×00.

5.9. EnCase

EnCase displays timestamps correctly if the *UTCOffset* fields have valid values. The assumption made by EnCase is that the *UTCOffset* field value 0×00 means UTC+0, but this is a wrong assumption. If the stored timestamp was stored using UTC+3, then the accuracy is 3 h off.

EnCase can be validated for interpreting exFAT timestamps when the *UTCOffset* fields contain valid values.

6. Conclusion and Further Work

- How do current exFAT implementations store timestamps?

In Windows 10 the exFAT specifications (Microsoft, 2021b) are followed by storing timestamps using the UTC offset of the local computer, including any daylight settings. MacOS has their own method of storing exFAT timestamps that switches the UTC sign and store the time accordingly. Linux Ubuntu 20.04 when using the exFAT fuse driver sets the *UTCOffset* fields to 0×00, which means the fields are not in use. Linux Ubuntu 20.04 native exFAT driver uses the *UTCOffset* fields, but sets them always to 0×80 (UTC+0). We also observed that graphical user interface apps could update the create timestamps in Linux to the modification time, or adjust it in MacOS making assumptions about the UTC offset previously registered.

- Can we use the UTC offset stored in a directory entry to describe the local time of the computer?

If the exFAT storage device has only been used on MacOS computers, we can switch the sign of the *UTCOffset* fields and find the local UTC offset used by the MacOS computer for a specific timestamp. If the storage device has only been used on a Windows computer, we can interpret the local UTC as equal to the *UTCOffset* field for a specific timestamp. However, if mix usage between Windows and MacOS then it may be more difficult. In Linux it is not possible to know what UTC offset were used, but still the local time is used for storing the timestamps when using the exFAT fuse driver, and UTC+0 when using the native exFAT driver.

We were not able to falsify our null hypotheses, because 33 percent of the valid *UTCOffset* observations in Experiment A showed that timestamps were stored using UTC+0, and only 67

percent were stored related to the local time. This means our main hypothesis is wrong for Linux Ubuntu 20.04 native exfat driver, but correct for the Windows and the MacOS exFAT driver.

- Do current forensic tools interpret exFAT timestamps differently?

The four different ways of storing exFAT timestamps between MacOS, Windows and Linux do impact tools that take the timezone *UTCOffset* fields into consideration (FTK Imager, X-Ways, and EnCase). Unfortunately, Autopsy does not consider the *UTCOffset* fields stored in the directory entry and uses the given timezone when adding the forensic image as the local time used for storing the timestamps. EnCase does not interpret exFAT with a non-valid *UTCOffset* field correctly, but make an assumption that the value 0×00 means UTC+0, which is incorrect in most cases. FTK Imager converts all timestamps to UTC+0 taking the *UTCOffset* fields into consideration. If one or more of the *UTCOffset* fields contains a non valid value, it only shows the timestamps for the valid *UTCOffset* fields. X-Ways take *UTCOffset* fields into consideration, and if these fields are all invalid it describes that local time (LT) is being used. However, X-Ways does not handle a mix of valid and invalid *UTCOffset* values, it then makes assumptions about the non-valid value in order to convert all timestamps of a file to the selected timezone.

It is not just the tools that may interpret exFAT differently, but also the different file system drivers may incorrectly change timestamps. MacOS makes an assumption that the created time uses the local time of the MacOS when the *UTCOffset* fields are invalid, and updates the Create time when changing a file using TextEdit by switching the UTC offset and storing the time accordingly. If the assumption is wrong, then the created time is stored incorrectly.

Finally, the conclusion is that exFAT timestamps stored by file systems may be unreliable, especially when used on multiple OSes, and that Digital Forensic tools may even interpret reliable dates in an unreliable way. We recommend using X-Ways or FTK Imager to interpret exFAT, and use patterns to identify which OS has been used in order to make an accurate interpretation of the timestamps.

As further work we suggest observing other file systems that can be used on multiple OSes, to assess if the drivers store timestamps equally, and if Digital Forensic tools interpret the timestamps accurately and reliably. Further, we recommend law enforcement to reassess criminal cases where exFAT and timestamps have been an important evidence to make sure innocent persons have not been convicted based on misinterpreted timestamps.

Acknowledgement

The research leading to these results has received funding from the Research Council of Norway programme IKTPLUSS, under the R&D project "Ars Forensica - Computational Forensics for Large-scale Fraud Detection, Crime Investigation & Prevention", grant agreement 248094/O70.

References

Bretel, J., 2017. Operating Systems and File Systems Compatibility. https://www.7dayshop.com/blog/operating-systems-and-file-systems-cross-compatibility-windows-apple-linux-playstation-xbox-android/.

Court of Human Rights, European, 2021. European Convention on Human Rights. https://www.echr.coe.int/Documents/Convention_ENG.pdf.

European Committee for Standardization, 2022. Cen Workshop Agrement (CWA): Requirements and Guidelines for a Complete End-To-End Mobile Forensic Investigation Chain. https://www.cencenelec.eu/media/CEN-CENELEC/CWAs/RI/cwa17865_2022.pdf.

Hamm, J., 2009. Extended Fat File System visited 2022-03-01. https://paradigmsolutions.files.wordpress.com/2009/12/exfat-excerpt-1-4.pdf.

Heeger, J., Yannikos, Y., Steinebach, M., 2021. Exhide: hiding data within the exfat file system. In: The 16th International Conference on Availability, Reliability and Security. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3465481.3470117. URL:

Ieong, R.S., 2006. Forza — digital forensics investigation framework that incorporate legal issues. Digit. Invest. 3, 29—36. https://doi.org/10.1016/j.diin.2006.06.004 the Proceedings of the 6th Annual Digital Forensic Research Workshop (DFRWS '06).

ISO, 2017. ISO/IEC 17025:2017 General Requirements for the Competence of Testing and Calibration Laboratories. https://www.iso.org/standard/66912.html.

Knight, J.C., Leveson, N.G., 1986. An experimental evaluation of the assumption of independence in multiversion programming. IEEE Transactions on Software Engineering SE- 12, 96—109. https://doi.org/10.1109/TSE.1986.6312924.

Lyle, J.R., 2010. If error rate is such a simple concept, why don't i have one for my forensic tool yet? Digit. Invest. 7, S135—S139. https://www.sciencedirect.com/science/article/pii/S1742287610000447. https://doi.org/10.1016/j.diin.2010.05.017. the Proceedings of the Tenth Annual DFRWS Conference.

Ma, G., Wang, Z., Cheng, Y., 2015. Recovery of evidence and the judicial identification of electronic data based on exfat. In: 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, pp. 66—71. https://doi.org/10.1109/CyberC.2015.10.

Microsoft, 2009. Us Patent Us 20090164440 A1. https://ppubs.uspto.gov/pubwebapp/, visited 2022-03-01.

Microsoft, 2021a. Default Cluster Size for NTFS, FAT, and exFAT. https://support.microsoft.com/en-us/topic/default-cluster-size-for-ntfs-fat-and-exfat-9772e6f1-e31a-00d7-e18f-73169155af95.

Microsoft, 2021b. exFAT File System Specification. https://docs.microsoft.com/en-us/windows/win32/fileio/exfat-specification.

Munegowda, K., Raju, G.T., Raju, V.M., 2012. Cluster allocation strategies of the exfat and fat file systems: a comparative study in embedded storage systems. In: Kumar, M., R. S., A., Kumar, T.V.S. (Eds.), Proceedings of International Conference on Advances in Computing. Springer India, New Delhi, pp. 691—698.

Munegowda, K., Raju, G., Raju, V.M., 2014. Directory compaction techniques for space optimizations in exfat and fat file systems for embedded storage devices. International Journal of Computer Science Issues (IJCSI) 11, 144.

Neale, C., Kennedy, I., Price, B., Yu, Y., Nuseibeh, B., 2022. The case for zero trust digital forensics. Forensic Sci. Int.: Digit. Invest. 40, 301352. https://doi.org/10.1016/j.fsidi.2022.301352. URL: https://www.sciencedirect.com/science/article/pii/S266628172200021X.

Nordvik, R., 2022. Exfat Forensic Images for Timestamp Testing. https://data.mendeley.com/datasets/krjsmdc65h/1, visited 2022-03-31.

Nordvik, R., Stoykova, R., Franke, K., Axelsson, S., Toolan, F., 2021. Reliability validation for file system interpretation. Forensic Sci. Int.: Digit. Invest. 37, 301174. https://doi.org/10.1016/j.fsidi.2021.301174. URL: https://www.sciencedirect.com/science/article/pii/S2666281721000822.

Scanlon, M., 2016. Battling the digital forensic backlog through data deduplication. In: 2016 Sixth International Conference on Innovative Computing Technology. INTECH), pp. 10—14. https://doi.org/10.1109/INTECH.2016.7845139.

Schullich, R., 2009. Reverse Engineering the Microsoft Extended FAT File System (exFAT). https://www.giac.org/paper/gcfa/570/reverse-engineering-microsoft-exfat-file-system/106672, visited 2022-03-01.

USB Memory Direct, 2022. Do I Need to Format a New USB Flash Drive? https://www.usbmemorydirect.com/blog/need-format-new-flash-drive/.

Vandermeer, Y., Le-Khac, N.A., Carthy, J., Kechadi, T., 2018. Forensic Analysis of the Exfat Artefacts, 08653 arXiv:1804.

Wani, M.A., AlZahrani, A., Bhat, W.A., 2020. File System Anti-forensics — Types, Techniques and Tools, vol. 2020. Computer Fraud & Security, pp. 14—19. https://doi.org/10.1016/S1361-3723(20)30030-0. URL: https://www.sciencedirect.com/science/article/pii/S1361372320300300.