2-28-2020

An Empirical study to determine the role of file-system in modification of hash value

Hash Value, New Technology File System (NTFS), Metadata, Master File Table, LogFile Sequence Number (LSN), Record Offsets

Follow this and additional works at: <https://vc.bridgew.edu/ijcic>

Part of the [Criminology Commons](#), [Criminology and Criminal Justice Commons](#), [Forensic Science and Technology Commons](#), and the [Information Security Commons](#)

Recommended Citation

Raychaudhuri, K., & Christopher, M. G. (2020). An empirical study to determine the role of file-system in modification of hash value. *International Journal of Cybersecurity Intelligence and Cybercrime*, 3(1), 24-41. <https://www.doi.org/10.52306/03010320NPTV7468>

This item is available as part of Virtual Commons, the open-access institutional repository of Bridgewater State University, Bridgewater, Massachusetts.

Copyright © 2-28-2020 Kumarshankar Raychaudhuri and M. George Christopher

Raychaudhuri, K. & Christopher, M.G. (2020). *International Journal of Cybersecurity Intelligence and Cybercrime*, 3 (1), 24-41.

An Empirical study to determine the role of file-system in modification of hash value

Kumarshankar Raychaudhuri*, LNJN National Institute of Criminology and Forensic Science,
Ministry of Home Affairs, Govt. of India, India

M. George Christopher, State Forensic Science Laboratory, India

Keywords; Hash Value, New Technology File System (NTFS), Metadata, Master File Table, LogFile Sequence Number (LSN), Record Offsets

Abstract:

In digital forensics, maintaining the integrity of digital exhibits is an essential aspect of the entire investigation and examination process, which is established using the technique of hashing. Lack of knowledge, while handling digital exhibits, might lead to unintentional alteration of computed hash, rendering the exhibit unacceptable in the court of Law. The hash value of a physical drive does not solely depend upon the data files present in it but also its file-system. Therefore, any change to the file-system might result in the change of the disk hash, even when the data files within it remain untouched. In this paper, our objective is to study the role of file-system in modification of the hash value. We examine and analyse the changes in the file-system of a NTFS formatted USB storage device, which leads to modification in its hash value when the device is plugged-in to the computer system without using write-blocker. The outcome of this research would justify the importance of write blockers while handling digital exhibits and also substantiate that the alteration in hash value of a storage device might not be an indication that data within the device has been tampered with.

Introduction

The initial phase of digital forensics examination starts with the acquisition of data from digital exhibits such as hard disks, USB storage devices etc. seized from the crime scene (Hosmer, 2002). This is done by creating a forensically sound copy of the storage media, which in turn, is achieved by creating a bit-stream image of it (Carlton, 2007). The essence, however, is to preserve the integrity of the exhibit and maintain a strict chain of custody (Kent et al., 2006; Kissel, 2013). As per (Vanstone et al., 1997) and the guidelines of the National Institute of Standards and Technologies (Kissel, 2013), Integrity of digital exhibits is defined as “the property whereby data within the exhibit has not been altered or modified in an unauthorized manner since the time it was created, transmitted or stored by an authorized source”, and is represented as hash value. To maintain integrity, forensic examiners use write-blocker to perform the imaging process, so as to prevent any unintentional changes to it by either the workstation or the operating system (Henry, 2009; Nelson et al., 2013).

*Corresponding author

Kumarshankar Raychaudhuri, LNJN National Institute of Criminology and Forensic Science, Ministry of Home Affairs, Govt. of India, New Delhi, India
Email: ksrc19@gmail.com

Reproduction, posting, transmission or other distribution or use of the article or any material therein, in any medium as permitted by written agreement of the International Journal of Cybersecurity Intelligence and Cybercrime, requires credit to the Journal as follows: “This Article originally appeared in International Journal of Cybersecurity Intelligence and Cybercrime (IJCIC), 2020 Vol. 3, Iss. 1, pp. 24-41” and notify the Journal of such publication.

© 2020 IJCIC 2578-3289/2020/02

With the advent of NTFS file-system, it has become the basis of pre-dominant operating systems in use, such as Windows XP, Windows Vista, Windows 7, Windows 8, 8.1 and Windows 10 (Newsham et. Al, 2007; Purcell & Lang, 2008; Svensson, 2005). Uniqueness of NTFS is its utilization of a journaling file system (NTFS, 1999), as a result of which, it keeps a record of all the changes taking place in a storage device or computer system by using a journal. This is done through MFT, which keeps data records of itself using metadata files (Rusbarsky & City, 2012). Metadata files are the files that describe the file-system and are instrumental in identifying whether an action is human or computer-oriented, thus revealing information about the creation, ownership, history, etc. of digital exhibits attached to the computer (Wee, 2006). Research (Kessler & Carlton, 2014) has been conducted for testing the scientific reason behind the requirement to use write-blockers during the acquisition of data from digital exhibits, especially in the case of NTFS formatted storage devices. Experiments have proved that the hash value of the storage device does get altered when no write-blocker is used. However, no literature or research experiment has provided a scientific explanation behind the alteration of the hash value of NTFS-formatted storage devices.

This research is undertaken with the objective of determining the role, the file-system plays in the modification of the hash value of a digital exhibit. In this direction, we try to establish the technical and scientific purpose behind the alteration of a hash value of any NTFS-formatted storage device even without any addition or deletion of data, when plugged in to a computer system in the absence of a write-blocker. To achieve this, an NTFS-formatted USB thumb drive is used in a Windows operating system. The thumb drive is plugged in and out of the computer system without using a write-blocker and the hash value is recorded through imaging by using a write-blocker. The process is repeated multiple times without any intentional addition or deletion of data to the disk. The MFT records from the images of the thumb drive in each instance are examined, analysed and compared using various tools and techniques to determine the changes taking place, which in turn lead to the variation in its hash value each time it is used in the absence of a write-blocker.

Literature Review

In NTFS, every entry is considered a file (Nelson et al., 2013; Rusbarsky, 2012). The first set of data is known as the Partition Boot Sector, starting at sector [0] and can expand up to 16 sectors. The Master File Table (MFT) lies immediately after sector [0] and covers around 12.5% of the disk (Nelson et al., 2013). The data (files and directories) in MFT are stored inside records, where each record is of the size 1024 bytes. The record fields consist of the metadata about the file or directory and the file's data or links to the file's data, which in turn depends on the size of the file. When the size of the file is less than 512 bytes, the data of the file along with its metadata is stored in the MFT record, known as the resident file. However, files larger than 512 bytes are stored outside the MFT, in external clusters, whose addresses are stored within the MFT records of the file. Such files are referred to as non-resident files (Svensson, 2005).

The first 15 records in the Master File Table are reserved for the system files, also referred to as metadata records. The metadata records in the MFT are as follows:

Table 1. Description of Metadata records of MFT in NTFS volume (Nelson et al., 2013)

Filename	System File	Record Position	Description
\$Mft	MFT	0	Base file record for each directory on the file-system; other record positions in MFT are allocated if more space is required.
\$MftMirr	MFT 2	1	It contains the first 4 records of MFT. In case of failure of a single sector of MFT, the records can be recovered, allowing restoration of MFT.
\$LogFile	Log File	2	NTFS stores the previous transactions in this position so that the system can be recovered in the event of a failure.
\$Volume	Volume	3	Information specific to the volume such as label and version, is stored in this location.
\$AttrDef	Attribute Definitions	4	This position consists of table listing the names, number and definitions of attributes
\$	Root filename index	5	This is the root folder on the NTFS volume.
\$Bitmap	Boot Sector	6	This file consists of an entire map of NTFS volume showing the clusters that are in use and those which are available.
\$Boot	Boot Sector	7	This is used to mount the NTFS volume during bootstrap process.
\$BadClus	Bad Cluster	8	This location gives information for clusters that have errors which cannot be recovered. An entry of the cluster location is made in this file.
\$Secure	Security File	9	This file contains the unique security descriptors for the NTFS volume. It also includes the access control list (ACL) for all files and folders on the NTFS volume.
\$Upcase	Upcase Table	10	It converts all lowercase characters to uppercase Unicode characters for the NTFS
\$Extend	NTFS Extension file	11	Optional extensions are listed in this file, such as quota, object identifiers etc.
		12-15	Reserved for future use

With the introduction of NTFS, the data storage technique by the operating system on the storage devices changed significantly (Alazab et al., 2009; Nelson et al., 2013). Not just user-data, but file-system metadata such as partition tables, MFT, etc. is also present in an NTFS formatted storage device. These metadata are used in the retrieval of information stored in a device and are usually managed by the operating system running on the host machine to which the device might be attached (Lyle, 2006).

Hashing plays a significant role in determining the authenticity and reliability of digital evidences (Cosic & Baca, 2010; Roussev, 2009). Hashing is defined as the technique wherein a fixed-length alphanumeric string is generated from a variable-sized input through hash algorithms (Kumar et al., 2012). The alphanumeric string is known as the Hash value or *Digest*. The hashing process forms the backbone of every digital forensic investigation. The hash value can be computed for a piece of data file or even for an entire exhibit such as a Hard Disk, USB devices, etc. The hash value of an exhibit is subject to change when new data contents are added to it, existing data contents are deleted from it or altered (Kumar et al., 2012), leading to a compromise in authenticity, reliability and integrity.

However, write-blocker plays a pivotal role in maintaining the integrity of a storage device, since the hash value also changes in its absence (Kessler & Carlton, 2014), even if there are no data addition, deletion or modification to the exhibit. In such an instance, it is extremely important to determine the different factors resulting in the change of drive hash value because this type of hash modification can be a deciding factor in digital forensic investigation.

Experimental Design

The experiment was performed with a USB thumb drive with a 4 GB capacity on a workstation having a 64-bit Windows 10 operating system, 8GB RAM and Intel(R) Core (TM) i7-7700 CPU @ 3.60GHz x-64 based processor.

Tools Used

The following tools were used for conducting the experiment:

- a. Tableau Forensic USB 3.0 Bridge – Tableau Forensic USB 3.0 Bridge is a hardware write-blocker, used during forensic acquisition (bit-stream imaging) of the USB thumb drive (Homeland Security, 2018).
- b. Tableau Forensic Duplicator – Forensic Duplicator is a modular multi-function device, which is used for both forensic imaging as well as cloning any storage media (Homeland Security, 2018). In this experiment, the tool was used for performing the wiping operation in the USB thumb drive.
- c. FTK Imager - Forensic Toolkit (FTK) Imager, is a disk-imaging tool used for bit-stream imaging of a physical drive, logical drive, CD/DVD, or contents of a file and folder (Homeland Security, 2016).
- d. NTFS File Extractor – NTFS File Extractor is a parsing tool, which extracts different files from the NTFS volume (Schicht, 2014). This tool was used for parsing the bit-stream image of the USB thumb drive for extraction of various metadata files.
- e. Mft2Csv – Mft2Csv is a parser, which is used for the extraction and decoding of \$MFT records information (Schicht, 2014).
- f. LogFileParser- LogFileParser is used for parsing the \$LogFile found in the NTFS volume (Schicht, 2014)
- g. Hex Editor – Hex Editor is a tool, which is used for viewing and editing the raw contents of disk drives, as well as to display and edit the memory used by the running processes. In this research, bit-stream image files with a size larger than 4GB, were analysed and examined using HxD (Horz, 2019).

Methodology

The following methodology was followed in carrying out the experiment:

1. A 4GB USB thumb drive is taken and wiped using Tableau Forensic Duplicator. It is formatted using NT file system on the workstation before starting the experiment.

2. A bit-stream raw image (NTFS1.001) of the NTFS formatted thumb drive is captured using FTK Imager. In this acquisition process, the thumb drive is connected to the workstation via a Tableau Forensic USB 3.0 Bridge (write-blocker). The hash value is noted.
3. Without adding any content to the thumb drive, it is plugged into the computer directly without using any write-blocker and plugged-out.
4. The imaging process in Step 2 is repeated and another raw image (NTFS2.001) is obtained. The hash value is again noted.
5. Steps 3 and 2 are repeated respectively, for obtaining a third raw image (NTFS3.001) of the thumb drive.
6. All the raw images (i.e., NTFS1.001, NTFS2.001, and NTFS3.001) are parsed using the tool NTFS File Extractor and the metadata files were extracted.
7. The \$MFT records, which are obtained from the raw images, are further parsed and decoded using the tool Mft2Csv. The parsed records are analysed and compared.
8. Also, the \$LogFile records are parsed using the tool LogFileParser and extracted files are analysed and compared.
9. The changes taking place in the \$MFT and \$LogFile, which in turn alter the hash value of the thumb drive, are compared, analysed and discussed in the next section.

Analysis of Results and Discussion

The plugging-in and plugging out of the NTFS-formatted USB thumb drive into the workstation without using write blocker, and subsequent imaging and hashing resulted in the following hash values (MD5 hash algorithm) in each of the instances, as given:

<i>Table 2. Acquired hash values of USB thumb drive in different instances of imaging</i>			
S.No	Name of image file of the thumb drive	Acquired MD5 hash value	Acquired SHA-1 hash value
1	NTFS1.001	ec48c53243d980ed2648d2115efe639a	bccb8d9f36078a209151dbf601eea31e5966d28f
2	NTFS2.001	b4db15b02b61e45fa48700aff8ed9163	35437db5520f9eefd7f02f1d9f5c01795ce4391e
3	NTFS3.001	918f9e97127034fee63d6b5e8dd468d8	4bbdd1cb834724b540d9214c84ef9f56e476fa3b

From Table 2, it can be seen that both MD5 and SHA-1 hash values of the thumb drive differ from one another everytime it was plugged-in and out of the workstation without a write-blocker.

Extraction of Metadata files from NTFS Volume

In order to examine, we analysed and compared the various changes taking place in the file-system of the storage media from one instance to another. The bit-stream imaging was done by attaching the media to the workstation through a write-blocker. On parsing the images by using NTFS File Extractor and FTK Imager, the following metadata files were obtained: \$Extend, \$AttrDef, \$BadClus, \$Bitmap, \$Boot, \$I30, \$LogFile, \$MFT, \$MFTMirr, \$Secure, \$TXF_DATA, \$Upcase, \$Volume

On examining and comparing the obtained metadata files, it was observed that changes took place only in \$MFT, \$LogFile and \$Extend files, while the rest of the metadata files remained unchanged in all the three images. To further determine the exact changes occurring in the above-mentioned metadata files, each of the files were parsed using a different software tool, extracting several other files from them.

The \$MFT files obtained from the images of the thumb drive were parsed, decoded and examined. It was observed that the MFT parser decoded the file at the “RecordOffset” level starting from 0x00000000 to 0x00040000 and data at an interval every 400 offsets are read. These 400 offsets consist of attributes such as “Signature”, “Integrity Check”, “HEADER_MFTRecordNumber”, “FN_Filename”, “FilePath”, “FileSizeBytes” and many more till “LOGGED_UTILITY_STREAM_ON”. The following file records for NTFS metadata files were found between RecordOffset 0x00000000 to 0x00009800 (Rusnovich et al., 2012):

It was also found that only till RecordOffset 0x00009800, the MFT file maintains file record data for a newly NTFS-formatted USB thumb drive, as shown in Fig. 2.

On comparing the parsed and decoded output of \$MFT files from all the images (NTFS1.001, NTFS2.001, and NTFS3.001), it was found that there was only a single change in the entire file (i.e., in the “Header_LSN” field, where LSN stands for LogFile Sequence Number) (Oh, 2013). LogFile Sequence Number (LSN) is a numerical value, which increases sequentially when there is a change in the file records for NTFS metadata files. The changes taking place in the field “Header_LSN” were shown in Fig. 3 and Fig. 4.

Figure 3. Snapshot showing the comparison of the \$MFT metadata file among NTFS1.001 and NTFS2.001

Figure 4. Snapshot showing the comparison of the \$MFT metadata file among NTFS2.001 and NTFS3.001

On examination and analysis of the images NTFS1.001 and NTFS2.001, it was found that at offset(h) 8008 from block(h) 8008-800F, the value of HEADER_LSN changed from 49 21 40 00 00 00 00 00 to 95 2E 40 00 00 00 00 00 (in Little endian format). The values were converted to Big Endian format and then to decimal format, which showed that the value of HEADER_LSN from “4202825” to “4206229”. Similarly, the value changes from “4206229” to “4206999” when the \$MFT files of images NTFS2.001 and NTFS3.001 were compared, examined, and analysed, as shown in Table 3.

Table 3. Comparison of HEADER_LSN value in different formats, parsed and decoded from \$MFT files acquired from different images of USB thumb drive

	NTFS1.001	NTFS2.001	NTFS3.001
Hexadecimal Little Endian	49 21 40 00 00 00 00 00	95 2E 40 00 00 00 00 00	97 31 40 00 00 00 00 00
Hexadecimal Big Endian	00 00 00 00 00 40 21 49	00 00 00 00 00 40 2E 95	00 00 00 00 00 40 31 97
Decimal	4202825	4206229	4206999

Analysis of \$LogFile Metadata File

The \$LogFile metadata files obtained from the images of the thumb drive were parsed, decoded, and examined, resulting in the extraction of 31 different files. In the analysis and comparison of the files, it was observed that only 19 files showed changed value of attributes from one image to another, as shown in Table 4.

Table 4. Comparative Analysis of Parsed and Decoded files from \$LogFile, among all three images of USB thumb drive

S.No	Name of the Parsed File	Changes in File
1	debug.log	Yes
2	LogFile.csv	Yes
3	LogFile.sql,	Yes
4	LogFile_AllTransactionHeaders.csv	Yes
5	LogFile_AttributeList.csv.empty,	No
6	LogFile_BitsInNonresidentBitMap.csv	No
7	LogFile_CheckpointRecord.csv	Yes
8	LogFile_DirtyPageTable32bit.csv.empty	No
9	LogFile_DirtyPageTable64bit.csv	Yes
10	LogFile_FileNames.csv	No
11	LogFile_INDX_I30.csv	Yes
12	LogFile_INDX_I30.sql	No

Continued on next page

The following changes were observed in different files extracted from the parsed \$LogFile:

Comparative analysis of LogFile_AllTransactionalHeaders.csv.. LogFile_AllTransaction-Headers.csv is involved in logging the records concerning “Redo pass” and “Undo pass” in the thumb drive. During “Redo pass”, NTFS redoes the “page update” records consisting of modifications to the volume (i.e., USB thumb drive) from the cache and flushes it to the disk. On the other hand, during “Undo pass”, non-committed transactions are rolled back to bring the system to a consistent state (www.cs.stju.edu.cn).

Table 4 – Continued from previous page

13	LogFile_IfUsnJrnl.csv.empty	No
14	LogFile_Mft_ObjectId_Entries.csv.empty	No
15	LogFile_ObjIdO.csv.empty	No
16	LogFile_OpenAttributeTable.csv	Yes
17	LogFile_QuotaO.csv	No
18	LogFile_QuotaQ.csv	No
19	LogFile_RCRD.csv	Yes
20	LogFile_ReparseR.csv.empty	No
21	LogFile_SecureSDH.csv	No
22	LogFile_SecureSII.csv	No
23	LogFile_SecurityDescriptors.csv	No
24	LogFile_SlackAttributeNamesDump.csv.empty	No
25	LogFile_SlackOpenAttributeTable.csv.empty	No
26	LogFile_TransactionTable.csv.empty	No
27	LogFile_TxfData.csv	No
28	LogFile_UpdateFileName_I30.csv	Yes
29	LogFile_UpdateFileName_I30.sql	Yes
30	MFTrecords.bin	No
31	ntfs.db	Yes

On analysis and subsequent comparison of LogFile.csv in all the three image files, records were logged, as shown in Table 5.

Table 5. Comparative Analysis of starting and ending record offsets of LogFile_AllTransactionHeaders.csv for all three image files (NTFS1.001, NTFS2.001 and NTFS3.001)

Image File	Starting Record Offset	Ending Record Offset	Total Offsets
NTFS1.001	0x00002040	0x00015F38	373
NTFS2.001	0x00004040	0x00015F38	383
NTFS3.001	0x00002040	0x00019DE0	416

The comparative analysis shown in Table 5 is graphically represented in Fig. 5 (See next page).

In the comparison and analysis of the graphs as shown in Fig. 5, the following observations were made:

- All three image files consist of 347 common records (marked in red color in Fig. 5) from offset 0x00004040 to 0x00015F38.
- 26 records (shown in blue color in Fig. 5) from offset 0x00002040 to 0x00003590 are logged only in NTFS1.001 and not logged in NTFS2.001.
- 36 new records (from offset 0x00016040 to 0x00017DF8) are created in continuation to the existing ones for the image file NTFS2.001, which are not present in NTFS1.001.
- 33 new records (from offsets 0x000025B0 to 0x00002DE0 and 0x00017F68 to 0x00019DE0) are found for the image file NTFS3.001

Figure 5. Comparative analysis of the number of offsets recorded in LogFile_AllHeadersTransactions.csv for the image files (NTFS1.001, NTFS2.001 and NTFS3.001) shown graphically

- The first record (at offset 0x00002040) in NTFS3.001 is similar to that of NTFS1.001. The next 5 records (from offsets 0x000025B0 to 0x00002DE0) are new ones, which are not logged in NTFS1.001 or NTFS2.001.
- Also, the last 27 records (from offset 0x00017F68 to 0x00019DE0) are newly logged in NTFS3.001 only.

On the basis of the differences found in the number of record offsets (as shown in Table 5), the existing “Redo/Undo operation” records in LogFile_AllTransactionHeaders.csv in NTFS1.001 were compared to the ones of the corresponding LogFile_AllTransactionHeaders.csv extracted from NTFS2.001, as shown in Table 6.

Table 6. Comparison of the records of “Redo/Undo Operations” in LogFile_AllTransactionHeaders.csv in NTFS1.001 and NTFS2.001

Hex Value	Redo Operation Record	Hex Value	Undo Operation Record	Number of operations in NTFS1.001	Number of operations in NTFS2.001
0x0E	AddIndexEntryAllocation	0x0F	DeleteIndexEntryAllocation	6	6
0x0C	AddIndexEntryRoot	0x0D	DeleteIndexEntryRoot	30	30
0x05	CreateAttribute	0x06	DeleteAttribute	29	29
0x06	DeleteAttribute	0x05	CreateAttribute	16	16
0x0F	DeleteIndexEntryAllocation	0x0F	AddIndexEntryAllocation	1	1
0x02	InitializeFileRecordSegment	0x00	Noop	24	24
0x25	JS_NewEndOfRecord	0x00	Noop	3	3
0x16	SetBitsInNonResidentBitMap	0x16	ClearBitsInNonResidentBitMap	24	24
0x0B	SetNewAttributeSizes	0x0B	SetNewAttributeSizes	36	36
0x09	UpdateMappingPairs	0x09	UpdateMappingPairs	4	4

Continued on next page

Table 6 – Continued from previous page

0x08	UpdateNonResidentValue	0x08	UpdateNonResidentValue	31	31
0x07	UpdateResidentValue	0x07	UpdateResidentValue	43	43
0x1E	AttributeNamesDump	0x00	Noop	2	3
0x01	CompensationLogRecord	0x00	Noop	4	6
0x1F	DirtyPageTableDump	0x00	Noop	2	3
0x1B	ForgetTransaction	0x01	CompensationLogRecord	49	50
0x1D	OpenAttributeTableDump	0x00	Noop	2	4
0x1C	OpenNonResidentAttribute	0x00	Noop	17	19
0x00	Noop	0x02	InitializeFileRecordSegment,	38	40
		0x01	CompensationLogRecord,		
		0x03	DeallocateFileRecordSegment,		
		0x00	Noop		
0x14	UpdateFileNameAllocation	0x14	UpdateFileNameAllocation	5	6
0x13	UpdateFileNameRoot	0x13	UpdateFileNameRoot	7	5

From Table 6, it is evident that there are nine records namely “AttributeNamesDump (0x1E), CompensationLogRecord (0x01), DirtyPageTableDump (0x1F), Noop (0x00), UpdateFileNameRoot (0x13), UpdateFileNameAllocation (0x14), OpenAttributeTableDump (0x1D), ForgetTransaction (0x1B) and OpenNonResidentAttribute (0x1C)” pertaining to “Redo/Undo operations” that are changing. The change in the number of operations from NTFS1.001 to NTFS2.001 results in the variation in the number of offsets (as mentioned in Table 5. Similar changes are also observed when a comparison is done between the LogFile_AllTransactionHeaders.csv of NTFS2.001 and NTFS3.001.

Comparative analysis of LogFile.csv. In the analysis and subsequent comparison of LogFile.csv in all the three image files, changes in logged records and record offsets were found, as shown in Table 7.

Table 7. Comparative Analysis of starting and ending record offsets of LogFile.csv for all three image files (NTFS1.001, NTFS2.001 and NTFS3.001)

Image File	Starting Record Offset	Ending Record Offset	Total Offsets
NTFS1.001	0x00002040	0x00015F38	373
NTFS2.001	0x00002058	0x00017DF8	401
NTFS3.001	0x00002040	0x00019DE0	418

The comparative analysis shown in Table 7 is graphically represented in Fig. 6 (See next page).

In the analysis and comparison of the graphs as shown in Fig. 6, the following observations were noted:

- The LogFile.csv in all three image files consist of 347 common records (i.e., from offset 0x00004040 to 0x00015F38) as marked in red color in Fig. 6.
- The first 26 records (marked in violet color in Fig. 6) logged from offset 0x00002040 till 0x00003590 in NTFS1.001 are different from the first 18 records (marked in blue color) logged from offset 0x00002058 till 0x00003D58 in NTFS2.001. It is also evident that the starting offset record in both the image files is different.

Figure 6. Comparative analysis of the number of offsets recorded in LogFile.csv for the image files (NTFS1.001, NTFS2.001 and NTFS3.001) shown graphically

- In NTFS3.001, the starting record offset is the same as that in NTFS1.001 (offset 0x00002040). However, the next 7 records (i.e., from offset 0x000025B0 to 0x000030E0) vary as compared to those of both NTFS1.001 and NTFS2.001
- The last 36 records in NTFS2.001 are also found in NTFS3.001 (marked in yellow color in Fig. 5) from offset 0x00016040 to 0x00017DF8.
- Image file NTFS3.001 consists of 27 newly logged records (marked in violet color in Fig. 5) from offset 0x00017F68 to 0x00019DE0, which are neither present in NTFS1.001 or NTFS2.001.

On the basis of the differences found in the number of record offsets (as shown in Table 7), the logged “Redo/Undo operation” records in LogFile.csv in NTFS1.001 were compared to the ones of the corresponding LogFile.csv extracted from NTFS2.001. Twenty one operations in 373 offsets (in NTFS1.001) and 383 offsets (in NTFS2.001) showed a match with those of LogFile_AllTransactionHeaders.csv (as shown in Table 6). There were an additional 18 “Redo/Undo operations” logged only in LogFile.csv of NTFS2.001, which were shown in Table 8.

Table 8. Additional “Redo/Undo operations” in LogFile.csv extracted from Image File NTFS2.001

Hex Value	Redo Operation	Hex Value	Undo Operation	Number of operations in NTFS2.001
0x1E	AttributeNamesDump	0x00	Noop	1
0x01	CompensationLogRecord	0x00	Noop	2
0x1F	DirtyPageTableDump	0x00	Noop	2
0x1B	ForgetTransaction	0x01	CompensationLogRecord	6
0x00	Noop	0x00	Noop	3
0x1D	OpenAttributeTableDump	0x00	Noop	1
0x14	UpdateFileNameAllocation	0x14	UpdateFileNameAllocation	1
0x07	UpdateResidentValue	0x07	UpdateResidentValue	2

The 18 Redo/Undo operations, which were logged only in LogFile.csv of NTFS2.001 increases the number of record offsets to 401 (as shown in Table 7) as compared to 383 offsets in LogFile_AllTransactionHeaders.csv of NTFS2.001, whereas both LogFile.csv and LogFile_AllTransactionalHeaders.csv have 373 record offsets in NTFS1.001 (as shown in Table 5 and 7). Changes in LogFile.csv also resulted in changes found in LogFile.sql.

Comparative analysis of LogFile_CheckpointRecord.csv. LogFile_CheckpointRecord.csv writes a checkpoint record, which describes the type of processing required to recover the NTFS volume if a crash occurs immediately. It also denotes the location in the \$LogFile to which NTFS must go in order to start the recovery process (www.cs.sjtu.edu.cn). The CheckpointRecord.csv file in NTFS1.001 consists of data written in four unique offset records. In the comparison to NTFS2.001, it is seen that new CheckpointRecord data is logged into three more offsets, making a total of seven offsets in the file. Further, it is also found that in NTFS3.001, another four CheckpointRecord data offsets are added, making a total of 11 offsets.

Comparative analysis of LogFile_DirtyPageTable64bit.csv. The LogFile_DirtyPageTable64bit.csv maintains the pages in the cache containing modifications to the file-system structure that have not yet been written to the disk. During recovery, the dirty pages must be flushed to the disk (Russovich et al., 2012). The DirtyPageTable64bit.csv file in NTFS1.001 consists of two records. In the comparison to NTFS2.001, it is seen that the new DirtyPageTable64bit contains five record offsets, giving a difference of three records in the file. Further, it is also found that in NTFS3.001, another three data offsets are added, making a total of eight records.

Comparative analysis of LogFile_RCRD.csv. LogFile_RCRD.csv denotes the records of the \$LogFile and is found at an interval of every 1000 offsets starting from the offset record 0x00002000. In the analysis of \$LogFile in NTFS1.001, it was seen that the offset records ended at 0x00015000, while the same ended at 0x00017000 and 0x00019000 for \$LogFile in NTFS2.001 and NTFS3.001 respectively, as shown in Fig. 6. Therefore, a comparison among the LogFile_RCRD.csv shows that data were logged into 2000 new record offsets every time imaging of the thumb drive was done without using a write-blocker.

Figure 7. A snapshot of the comparative analysis of the record offsets of LogFile_RCRD.csv extracted from parsing \$LogFile from NTFS1.001, NTFS2.001 and NTFS3.001, using Hex Editor

Comparative analysis of LogFile_INDX_I30.csv. Analysis of LogFile_INDX_I30.csv for NTFS-1.001 and in NTFS2.001 shows the addition of one offset record 0x00008928 in the latter. However, in the comparison of LogFile_INDX_I30.csv in NTFS2.001 and NTFS3.001, the offset record 0x00008928, which was added in NTFS2.001, could not be found in NTFS3.001.

Comparative analysis of LogFile_UpdateFileName_I30.csv. The file LogFile_UpdateFileName_I30.csv consists of all the decoded UpdateFileNameRoot and UpdateFileNameAllocation for both redo and undo operations (lib4dev.in). UpdateFileNameRoot and UpdateFileNameAllocation were the undo and redo operation records, which were contained in the file LogFile_AllTransactionHeaders.csv.

In comparing all three LogFile_UpdateFileName_I30.csv, it was observed that they contained an equal number of record offsets (i.e., 24 record offsets). While examining and analysing all the parsed data, it was inferred that out of the 24 record offsets, 12 were redo operations (If_IsRedo Flag set to 1), while the rest of the records were the corresponding undo operations (If_IsRedo Flag set to 0). In NTFS1.001, the record offsets start from 0x00002098 and end at 0x00015F38. However, on comparing it to NTFS2.001, it was observed that the record offsets start from 0x00003058 and end at 0x00017058. Record offset 0x00004870 till 0x00015F38 were found to be contained in both NTFS1.001 and NTFS2.001 (i.e., eight offsets are similar in both the disk images, whereas four offsets, which are logged in NTFS2.001 are not found in the former). Similarly, in NTFS3.001, the same eight records were found, starting from 0x00004870 and ending at 0x00018868. In the comparison of disk images NTFS2.001 and NTFS3.001, three more records were found to be matching, (i.e., from offset number 0x00016098 till 0x00017058), while only an extra record offset 0x00018868 was logged in the latter.

Comparative analysis of LogFile_OpenAttributeTable.csv. The file LogFile_OpenAttributeTable.csv in NTFS1.001 consists of 19 unique record offsets starting from 0x00002798 till 0x00014868. When compared to the corresponding counterpart in NTFS2.001, it was observed that it consisted of 22 unique records starting from offset 0x000040E0 and ending at 0x00017778. On further comparison with that of NTFS3.001, there was a difference of six records. In NTFS3.001, the records start from the offset 0x00002800 and terminate at 0x00019800. Analysis of LogFile_OpenAttributeTable.csv in all the three bit-stream images revealed that there were 18 record offsets from 0x000040E0 till 0x00014868, which were common in all the images. Although, there was a difference of six offsets between NTFS2.001 and NTFS3.001, three records (i.e., from 0x00016798 to 0x00016F90) have been logged in both image files.

Analysis of \$Tops metadata file

\$Tops is a file present inside the \$Extend metadata file, which is a directory consisting of the extended metadata file. The file is present in the location \$Extend \$RmMetadata \$TxfLog \$Tops. TxF Old Page Streams (TOPS) is a file, which stores data that has been overwritten inside a currently active transaction (Metz, 2019). The \$Tops file was compared to the images NTFS1.001, NTFS2.001, and NTFS3.001. The value of offset 0x000000020 changed from 01 in NTFS1.001 to 00 in NTFS2.001, as shown in Fig.8. However, no change was observed in the \$Tops file when the image file NTFS2.001 was compared to NTFS3.001.

Figure 8. A snapshot of the comparison of contents of \$Tops file extracted from the parsing of NTFS1.001 and NTFS2.001, using Hex Editor

Conclusion and Suggestions

During the experiments performed the comparison and analysis of the various metadata files of NTFS, extracted by parsing and decoding bit-stream images of the USB thumb drive was done. It was observed that there were significant changes in the file-system. The major changes were found in metadata files namely \$MFT, \$LogFile and \$Tops. In \$MFT file, there was a change in the values recorded for LogFile Sequence Number (LSN). While parsing and decoding \$LogFile, 31 files were generated. Analysis of these 31 files revealed changes in the contents of 12 files namely debug.log, LogFile.csv, LogFile.sql, LogFile_AllTransactionHeaders.csv, LogFile_DirtyPageTable64bit.csv, LogFile_INDX_I30, LogFile_OpenAttributeTable.csv, LogFile_RCRD.csv, LogFile_UpdateFileName_I30.csv, LogFile_UpdateFileName_I30.sql, LogFile_CheckpointRecord.csv, and ntfs.db. Significant modifications were found to exist in the 12 files parsed from \$LogFile. The third metadata file also showed a minute alteration in the \$TOPS (TxF Old Page Streams) file. In \$Tops file, the change was observed only when a comparison was done among NTFS1.001 and NTFS2.001, while no change was taking place when NTFS2.001 was compared to NTFS3.001.

Thus, after performing the experiments in this research, it was concluded that the NTFS file-system plays a major role in changing of hash value of a digital storage device such as the hard disk drive, USB disks, etc. The hash value of the storage device is liable to get modified even when the device has been freshly wiped and formatted, and it is used without a write-blocker. The hash value of a storage device might not always change due to a change in the file contents of the device. This research portrays the importance and essence of write-blockers during the examination of digital evidence. Based on the conclusions, the following suggestions are given:

- More significance should be given to the hash value of individual data files contained in an exhibit rather than that of the entire exhibit, during the course of investigation. The hash values of all the individual data files present inside the exhibit should strictly be computed and documented.
- The hash value of the exhibit might get altered accidentally due to the absence of write-blockers during the acquisition or examination phase, whereas the hash value of the indivi-

dual data files will remain unaltered. Under such circumstances, it should not be assumed or interpreted that there is tampering done to the individual files within the exhibit.

- After dispatch of the digital exhibit from the forensic laboratory, later in the courtroom, if the computed hash value of the exhibit does not match the documented hash value, in such circumstances, the hash values of the individual files should be matched to check whether any actual tampering has been done or not.
- Computing the block hash values of the exhibit as compared to the entire exhibit should be considered as a best practice.

The outcome of this research would be extremely useful and helpful for the forensic fraternity in dealing with the investigation of cyber-crime cases, especially when disk forensics is to be done.

Limitations and Future Scope

This research has been conducted with the objective of determining the changes that take place inside the NTFS file-system of a storage device, which leads to a change in the hash value of the device when the storage device is plugged-in and plugged-out of the computer system without making any addition, deletion or modification of data to it. After performing the experiments on a USB thumb drive, it was concluded that several changes take place in the Master File Table, which is part of the NTFS file-system of the device. However, there have been some limitations while carrying out the research. The scope of our research has given rise to few limitations or constraints, which can be resolved in the future by carrying out similar studies.

The following are the limitations and future scopes of the study conducted:

- a) In this research, our scope is limited to carrying out the experiments on USB thumb drive with NTFS file-system. However, the same experiment can also be carried out on other storage devices like Hard Disks, Solid-State Drives (SSDs) etc. with NTFS file-system.
- b) Only a single file-system has been considered in this experiment due to its wide and popular usage in the Windows operating system. In the future, other different file-systems such as FAT32, Ext, HFS, etc. can also be experimented upon using a similar methodology and a comparative analysis can be done among the results.
- c) The work performed in this study is independent of time factor. The plug-in and plug-out time of the storage devices have not been taken into account. However, similar work can be conducted in the near future with a dependence on time factor to determine if any pre-dominant changes are there.

Declaration of Interest Statement

The authors declare that they have no conflicts of interest.

References

- Alazab, M., Venkatraman, S., & Watters, P. (2009). Effective digital forensic analysis of the NTFS disk image. *Ubiquitous Computing and Communication Journal*, 4(3), 551- 558.
- Carlton, G.H. (2007). *A protocol for the forensic data acquisition of personal computer workstations*. Ann Arbor, MI, ProQuest.
- Ćosić, J., & Bača, M. (2010, May). (Im) proving chain of custody and digital evidence integrity with time stamp. In *The 33rd International Convention MIPRO* (pp. 1226-1230).

- Henry, P. (2009). Best practices in digital evidence collection. SANS. Retrieved from <https://www.sans.org/blog/best-practices-in-digital-evidence-collection/>
- Homeland Security Science and Technology. (2016). FTK Imager v3.4.2.6 - Test Results for Disk Imaging Tool. Retrieved from https://www.dhs.gov/sites/default/files/publications/1491_508_Test\%20Report_NIST_Disk\%20Imaging_FTK\%20Imager\%20v3.4.2.6_October_14_2016.pdf
- Homeland Security Science and Technology. Tableau TD3 forensic imager v2.0.0, test results for disk imaging tool- federated testing suite. Retrieved from https://www.dhs.gov/sites/default/files/publications/508_Test\%20Report_NIST\%20Disk\%20Imaging\%20Tool\%20Tableau\%20TD3\%20Forensic\%20Image\%20v2.0.0\%20August\%202018_Final.pdf
- Homeland Security Science and Technology (2018). Tableau Forensics USB 3.0 Bridge T8u , Test Results for Disk Imaging Tool- Federated testing Suite. Retrieved from https://www.dhs.gov/sites/default/files/publications/Test\%20Report_NIST_HWB_Tableau\%20Forensic\%20USB\%203.0\%20Bridge\%20T8u_Firmware\%20Version\%20Apr\%2026\%202018\%2008.49.42_October\%202018.pdf
- Hosmer, C. (2002). Proving the integrity of digital evidence with time. *International Journal of Digital Evidence*, 1(1), 1-7.
- Kent, K., Chevalier, S., Grance, T., & Dang, H. (2006). Guide to integrating forensic techniques into incident response. Retrieved from <https://csrc.nist.gov/publications/detail/sp/800-86/final>
- Kessler, G. C., & Carlton, G. H. A study of forensic imaging in the absence of write-blockers. (2014). *The Journal of Digital Forensics, Security and Law*, 9(3), 51-58.
- Kissel, R., Scholl, M., Skolochenko, S., & Li, X. (2013). Guidelines for media sanitization. Retrieved from <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-88r1.pdf>
- Kumar, K., Sofat, S., Jain, S. K., & Aggarwal N. (2012). Significance of hash value generation in digital forensic: A case study. *International Journal of Engineering Research and Development*, 2(5), 64-70.
- Lyle, J. R. (2006). A strategy for testing hardware write block devices. *Journal of Digital Investigations*, 1(3), 3-9.
- Mael, H. (2019). HxD- Freeware hex editor and disk editor. Retrieved from www.mh-nexus.de/en/hxd
- Meffert, C. S., Baggili, I., & Breitingner, F. (2016). Deleting collected digital evidence by exploiting a widely adopted hardware write-blocker. *Journal of Digital Investigations*, 7(18), S87-96.
- Metz, J. (2019). New Technologies File System (NTFS). Retrieved from [https://github.com/libyal/libfsntfs/blob/master/documentation/New\%20Technologies\%20File\%20System\%20\(NTFS\).asciidoc](https://github.com/libyal/libfsntfs/blob/master/documentation/New\%20Technologies\%20File\%20System\%20(NTFS).asciidoc)
- Mikhailov, D. (2019). NTFS file system. Retrieved from <http://itc.upt.al>
- Nelson, B., Phillips, A., & Steuart, C. (2013). *Guide to Computer Forensics and Investigations* (4th ed.). Boston, MA: Cengage Learning.
- Newsham, T., Palmer, C., Stamos, A., & Burns, J. (2007). Breaking forensics software: Weaknesses in critical evidence collection. Retrieved from <https://pdfs.semanticscholar.org/da89/2ddca1a625bfac3fe8f5b32447004db16eca.pdf>
- NTFS, Active@ Data recovery Software. Retrieved from <http://www.ntfs.com>

- Oh, J. (2013). *NTFS Log Tracker*. Forensic Insight: Digital Forensics Community in Korea.
- Parser for \$LogFile on NTFS. Retrieved from <https://devhub.io/link>
- Purcell, D., & Lang, S. (2008). Forensic artifacts of Microsoft windows vista system. In Yang C.C. et al. (Ed.), *Intelligence and Security Informatics. ISI 2008* (pp. 304-319). Springer, Berlin, Heidelberg: Springer.
- Rusbarsky, K. L., & City, K. (2012). A forensic comparison of NTFS and FAT32 file systems. Retrieved from https://www.marshall.edu/forensics/files/RusbarskyKelsey_Research-Paper-Summer-2012.pdf
- Russinovich, M., & Solomon, D. A. (2012). *Ionescu A. Windows Internals – Part 1* (6th ed.). Redmond, Washington: Microsoft Press.
- Russinovich, M., & Solomon, D. A. (2012). *Ionescu A. Windows Internals – Part 2* (6th ed.). Redmond, Washington: Microsoft Press.
- Schicht, J. (2014). NTFS File Extractor. Retrieved from <https://github.com/jschicht/NtfsFileExtractor>
- Schicht, J. (2014). Mft2Csv. Retrieved from <https://github.com/jschicht/Mft2Csv>
- Schicht, J. (2014). LogFileParser. Retrieved from <https://github.com/jschicht/LogFileParser>
- Svensson, A. (2005). *Computer forensic applied to windows NTFS computers*. Stockholm's University, Royal Institute of Technology.
- Roussev, V. (2009). Hashing and data fingerprinting in digital forensics. *IEEE Security and Privacy*, 7(2), 49-55.
- Vanstone, S., van Oorschot, P., & Menezes, A. (1997). *Handbook of applied cryptography*. Boca Raton, FL: CRC Press.
- Wee, C. K. (2006). Analysis of hidden data in NTFS file system. Retrieved from <https://www.forensicfocus.com/downloads/ntfs-hidden-data-analysis.pdf>