©2018 Published in 6th International Symposium on Innovative Technologies in Engineering and Science 09-11 November 2018 (ISITES2018 Alanya – Antalya - Turkey)

Recovering Data Using MFT Records in NTFS File System

[1]Süleyman Gökhan TAŞKIN and [2]Ecir Uğur KÜÇÜKSİLLE
[1]Department of Information Technology, Bandirma Onyedi Eylul University, Turkey
[2]Faculty of Engineering, Department of Computer Engineering, Suleyman Demirel University, Turkey

Abstract

Data storage devices use a specific structure when storing or accessing the stored data. This is called file system. Before beginning to store data in the data storage device, it must be formatted absolutely. While this data storage device is being formatted, the file system should be selected.
NTFS, the most commonly used file system, keeps the files in the disk as a list in the MFT (Master File Table) file. Even if the file is deleted, the file record in this table will not be deleted. The physical location of the file can be found by looking at these MFT records.
In this study, computer software was created on the basis of restoring the disk using a MFT file of the NTFS file system, and the result was examined.
When national studies are examined, data recovery programs on the market are compared with each other. When international studies are examined, it is seen that NTFS and MFT concepts are explained but data recovery method using MFT records is not examined in detail.

Key words: NTFS file system, cluster, partition, MFT, data recovery, forensic analysis

1. Introduction

Today, it is observed that many institutions and individuals prefer digital environment as data storage method. The dizzying developments in the area of cloud computing are also a sign that this situation will grow even further.
Personal information and documents are stored in many electronic devices such as computers, laptops, tablets, phones, portable disks. The common point of these devices is the presence of a storage unit of all. Stored documents and information are stored on these storage devices. Data loss can also be caused by the corruption of these storage devices, or by deleting the file to prevent accidental or unauthorized access.
Data loss caused by storage device corruption can be referred to as hardware data loss. Hardware failure; such as head failures, engine failures, deformation on the disk surface, scratches on the disk surface, bad sector formation, integrated circuit failure, loss of functionality due to contamination of the read / write heads, sticking to the disk surface of the read / write head [1].
Data loss is caused by the user's intentional or unintentional deletion of files, disk formatting of the user, corruption of the file system, etc. It occurs for reasons.
Data recovery is the process of ensuring that data stored by people, institutions or corporations in the digital environment becomes inaccessible due to the above-mentioned data loss. Data loss due to storage device corruption can be recovered by disk intervention with special tools and special media. To recover data in the case of software data loss, the storage device must be in a working state. Data recovery can be done through software assistant software data loss.

Corresponding author: Address: Bandırma Onyedi Eylül University, Faculty of Information and Technology, Balıkesir/TURKEY

Naiqi et al. (2008), in their study, a data recovery tree for deleted files and their application has been compared with the commercially available EasyFileRecovery program [2].
Mahant and Meshram (2012) have developed an application for data recovery from USB devices [3].
Ravindra et al. (2013) stated that the data recovery process is directly related to disk size, and they have developed an algorithm for faster data recovery [4].
When national studies in the literature are examined, the existing paid or unpaid practices are compared and their performances are evaluated. In the case of national academic studies, we mentioned hardware data recovery but no study on software-based data recovery.
When international academic studies are examined, the NTFS file system and MFT structure are mentioned but it is not mentioned how to recover the data in detail.
When the existing applications are examined, data recovery method is presented as fast search method using MFT file, and each application has its own different algorithms.
In this study, the data recovery method using MFT file was investigated in detail in the software data loss and the application of this method was investigated and the data recovery was researched in performance so that this deficiency in the literature was tried to be filled.
Forensic analysis is an important factor in the confidentiality of data, so it is important to recover data with national applications. The aim of this study is to provide a basis for national studies on data recovery and to increase the studies in data recovery.

2. Materials and Method

When a data storage device is desired to be formatted, it is necessary to select which file system this storage device is to be formatted with. File systems are systems that determine which scheme of data is stored on the storage device. The formatting process makes the disk partition fit the selected file system so that the stored files are kept in the selected file system order.
The NTFS (New Technology File System) file system is a new file system developed by Microsoft to eliminate the limitations of FAT file systems. This file system has features that are not found in the FAT file system, such as multi-user, user authorization, user quotas on disk.
The NTFS file system hosts files in a cluster, and has a table that keeps track of which clusters these files are in. The clusters are independent of the sections. This reduces the fragmentation of files on the disk, resulting in both efficient use of space and increased performance, especially on high-capacity hard drives. The largest 2 GB files can be stored in the FAT file system. With the FAT32 file system, this size has been increased to 4 GB. Since the file sizes are so large today, the maximum size of a single file in the NTFS file system is 16 TB [5].
The most important features of the NTFS file system are security-related features. Thanks to this file system; Quotas can be set on disk for users, which users can access which files or folders can be defined by the system administrator. Active Directory and Domain Controller features are the security features that come with the NTFS file system.
It supports long file names and file names originating from Unicode. Two areas on the disk are very important during the boot process. In these areas, the "Master Boot Record (MBR)" and the other is "Volume Boot Record (VBR)". The MBR is on disk 0th sector, 0th cylinder and 0th head. It is not part of any partitions. VBR is found in sector 0 of each section.

2.1. Master Boot Record (MBR)

The MBR is found in the first sector of the physical disk and identifies the partitions on the disk. Occurs when the disk is partitioned. The MBR stores the boot code and partition table used by the BIOS to read the partitions. The last two bytes of the MBR are set to 55 AA. The disk is located in the offset 0x01B8, which is a unique number required to identify the operating system to the operating system [6].
The boot code scans the boot table that identifies the partitions on the disk to find the active partition, finds the boot sector of this boot partition, and loads the VBR in that boot partition.
The 0x1BE offset of the partition table in the MBR is used to specify the locations of partitions on the disk and the partition type. This data structure is explained in the Table 1.

Table 1. MBR Structure [7]

Byte Range	Description
0x0 – 0x0	Bootable flag: 0x00 is standart, 0x08 is bootable
0x1 – 0x3	Starting CHS Address
0x4 – 0x4	Partition Type: 0x07 is NTFS

2.2. Volume Boot Record (VBR)

VBR is found in the zeroth logical sector in the active partition. While VBR is located in the first sector, there is also an operating system installer in the immediate sectors.
During PC boot or reboot processes, the CPU registers specify the EIP (Extended Instruction Pointer) address that holds the standard values and the code addresses that are executed by the CPU. This allows the transition to the BIOS start point. After the power-on-self-test (POST), the BIOS is located in the MBR located in the discrete zero sector. The BIOS loads the MBR into memory (RAM) and runs the MBR boot code that finds the boot partition by scanning the partition table. The MBR loads the operating system kernel and loads the VBR of the boot partition that will complete booting and completes the boot process.
When the NTFS partition is installed, the partition contains $ AttrDef, $ BadClus, $ Bitmap, $ Boot, $ LogFile, $ MFT, $ MFTMir, $ Secure, $ UpCase, and $ Volume meta files. $ AttrDef, $ BadClus, $ Bitmap, $ Boot, $ LogFile, $ MFT, $ MFTMir, $ Secure, $ UpCase and $ Volume metadata.
The 16 sectors in the start section contain the VBR and NTLDR (bootstrap) code. This partition is located in the "$ Boot" file in NTFS. The "$ Boot" file can be accessed by the following Table 2.

Table 2. $BOOT metadata file [8]

Byte Range	Description
0x00 – 0x02	Jump Instruction.
0x03 – 0x0A	OEM ID.
0x0B – 0x0C	Bytes Per Sector.
0x0D – 0x0D	Sectors Per Cluster.
0x0E – 0x0F	Reserved Sectors.

0x10 – 0x12	Always 0.
0x15 – 0x15	Media Descriptor.
0x16 – 0x17	Always 0.
0x18 – 0x19	Sectors Per Track.
0x1A – 0x1B	Number of Heads.
0x1C – 0x1F	Hidden Sectors.
0x28 – 0x2F	Total Sectors.
0x30 – 0x37	Logical Cluster Number for the file $MFT.
0x38 – 0x3F	Logical Cluster Number for the file $MFTMirr
0x40 – 0x43	Clusters Per File Record Segment.
0x44 – 0x47	Clusters Per Index Block.
0x48 – 0x49	Volume Serial Number.
0x50 – 0x53	Checksum
0x54 – 0x1AA	Bootstrap Code.
0x1FE – 0x1FF	0xAA55 MBR Signature

2.3. Volume Boot Record (VBR)

Everything in a NTFS file system is a file. All files are kept in the MFT table. There are one or more records in the MFT file for all files and folders. The MFT table consists of records with a size of 1024 bytes.

The entries in the MFT cannot be deleted at all, even if the file or folder is deleted. When the file or folder is deleted, the MFT record is marked as deleted. Even if the file is deleted on this point, it can be found and used again.

In the MFT records, the MFT file contains information about all the files, including their own information. When examining the MFT record, the files in the partition begin with the metadata prefix files starting with the "$" sign. The metadata files of the MFT file are described in Table 3.

Table 3. Metadata Files of the MFT [7]

Byte Range	Description
$MFT	Contains one base file record for each file and folder on an NTFS volume.
$MFTMir	Guarantees access to the MFT in case of a single-sector failure. It is a duplicate image of the first four records of the MFT.
$LogFile	Contains information used by NTFS for faster recoverability.
$Volume	Contains information about the volume, such as the volume label and the volume version.
$AttrDef	Lists attribute names, numbers, and descriptions.
.	The root folder.
$Bitmap	Represents the volume by showing free and unused clusters.
$Boot	Includes the BPB used to mount the volume and additional bootstrap loader code used if the volume is bootable.
$BadClus	Contains bad clusters for a volume.
$Secure	Contains unique security descriptors for all files within a volume.
$Upcase	Converts lowercase characters to matching Unicode uppercase characters.

$Extend	Used for various optional extensions such as quotas, reparse point data, and object identifiers.
$Quota	Used for disk quotas.
$ObjId	Used for distributed link tracking.
$Reparse	Used for reparse points.

All MFT records have the same structure. This structure is 1024 bytes in size. The first 48 byte header field, the next 8 bytes of correction data, and the next one contains file attributes. The header section of the first 48 bytes defines the attributes of the entry. Attribute data, after the header field and correction data, contains a lot of information from the file name to the file data [9]. The reference number is used to associate file records, folders, and MFT records with each other.
While the attributes of MFT files are often contained within the MFT record, some attributes rarely fit into the MFT record. In this case, the file attributes can be found outside the MFT record and in a different cluster. These additional records in a different cluster hold the reference number of the main MFT record at offset 0x20. If the attribute is in the MFT record, this value is zero. The attributes contained in the MFT record are called the resident attributes, and the attributes that are not in the MFT record and are in a different cluster are called non-resident attributes.

2.3.1. Attribute Header

It has been mentioned that the attributes are divided into two parts: the resident in the MFT record and the non-resident record in the MFT record. Resident attributes and non-resident attributes also differ in the title sections.

Table 4. Resident and Non-Resident Attributes [10]

Resident Attributes		Non-Resident Attributes	
0x00-0x03	Attribute Type	0x00-0x03	Attribute Type
0x04-0x07	Length	0x04-0x07	Length
0x08-0x08	Non-resident flag	0x08-0x08	Non-resident flag
0x09-0x09	Name length	0x09-0x09	Name length
0x0A-0x0B	Offset to the Name	0x0A-0x0B	Offset to the Name
0x0C-0x0D	Flags	0x0C-0x0D	Flags
0x0E-0x0F	Attribute Id	0x0E-0x0F	Attribute Id
0x10-0x13	Length of the Attribute	0x10-0x17	Starting VCN
0x14-0x15	Offset to the Attribute	0x18-0x1F	Last VCN
		0x20-0x21	Offset to the Data Runs
		0x22-0x23	Compression Unit Size
		0x24-0x27	Padding
		0x28-0x2F	Allocated size of the attribute
		0x30-0x37	Real size of the attribute
		0x38-0x3F	Initialized data size of the stream

2.3.2. Standard Information Attribute

This attribute stores information about updating and modifying the file. Table V explains the offsets and values of the "$STANDARD_INFORMATION" attribute.

Table 5. Standard Information Attribute Offsets [11]

Offset	Description
0x00 – 0x07	C Time - File Creation Time
0x08 – 0x0F	A Time - File Altered Time
0x10 – 0x17	M Time - MFT Changed Time
0x18 – 0x1F	R Time - File Read Time
0x20 – 0x23	DOS File Permissions
0x24 – 0x27	Maximum Number of Versions
0x28 – 0x2B	Version Number
0x2C – 0x2F	Class Id
0x30 – 0x33	Owner Id
0x34 – 0x37	Security Id
0x38 – 0x3F	Quota Charged
0x40 – 0x47	Update Sequence Number (USN)

2.3.3. File Name Attribute

The "$FILE_NAME" attribute keeps the file name in Unicode-16 encoding and shows the creation time and modification time of the file as well as the $STANDARD_INFORMATION attribute. The "$FILE_NAME" attribute also shows the size and actual size of the file on the disk. The disk size field shows the total number of clusters the file has on the disk.
Table 6 explains the offsets and values of the "$FILE_NAME" attribute.

Table 6. File Name Attribute Offsets [8]

Byte Range	Description
0x00 – 0x07	File Reference to parent directory.
0x08 – 0x0F	File creation time.
0x10 – 0x17	File modification time.
0x18 – 0x1F	MFT modification time.
0x20 – 0x27	File access time.
0x28 – 0x2F	Allocated size of file.
0x30 – 0x37	Real size of file
0x38 – 0x3B	Flags
0x3C – 0x3F	Used by EAs and Reparse
0x40 – 0x40	Filename length in unicode characters
0x41 – 0x41	Filename namespace
0x42 – 0x42	File Name in Unicode

2.3.4. Data Attribute

In previous sections, "resident" and "non-resident" attributes were explained. If the file is "resident", the data of this file is kept "$ DATA". For files with "non-resident" attribute, the starting number and size of the cluster containing this file are kept in "$ DATA" attribute. If the offset of 0x08 of "$DATA" attribute is 01, it is "non-resident" file and 00 is "resident" file.
DataRun values are used to indicate the location of "non-resident" file fragments. In this datarun values, the first cluster number of the file part and the cluster count of the file part are stored. Datarun consists of 3 parts. These; the length and offset value in 1 byte size, the number of datarun clusters, and the cluster start number. The length and offset value of 1 byte size, the first 4 bits indicate the count of datarun clusters, and the other 4 bits indicate the cluster start number [12].

2.3.5. Attribute List Attribute

The $ ATTRIBUTE_LIST attribute holds a list of attributes in the MFT record. This attribute can be resident in the MFT record or non-resident in the MFT record. Table VII explains the attribute types.

Table 7. Attribute List Attribute Offsets [13]

Byte Range	Description
0x10	$STANDARD_INFORMATION
0x20	$ATTRIBUTE_LIST
0x30	$FILE_NAME
0x40	$VOLUME_VERSION
0x50	$SECURITY_DESCRIPTOR
0x60	$VOLUME_NAME
0x70	$VOLUME_INFORMATION
0x80	$DATA
0x90	$INDEX_ROOT
0xA0	$INDEX_ALLOCATION
0xB0	$BITMAP
0xC0	$SYMBOLIC_LINK
0xD0	$EA_INFORMATION
0xE0	$EA
0xF0	$PROPERTY_SET
0x100	$LOGGED_UTILITY_STREAM

If an MFT record has too many attributes, or if the size of the attributes is large, the size can exceed 1024 bytes. For this reason, the attributes do not fit into a single MFT record and can be kept in more than one MFT record. Attribute information stored in multiple records is also specified in the $ATTRIBUTE_LIST attribute.
The size of the attributes listed in the $ ATTRIBUTE_LIST attribute can be large enough to fit in a single MFT record. In this case, the non-resident flag of the $ ATTRIBUTE_LIST attribute is "01" and the attribute list is stored on another cluster.

2.4. Data Recovery By MFT File

Even if the files on the MFT files on the disk are deleted, the file's location on the disk, file size and file name etc. the attributes will continue to be stored.
Data recovery using the MFT file results in faster results than data recovery management by moving individual clusters of a disk one by one. If a partition's MFT file is intact and still readable, it will be more convenient to find the file by scanning the MFT record for deleted files.
Before you can recover data with this method, the file's MFT record is found. The file containing the MFT record is listed with the $ATTRIBUTE_LIST attribute and the MFT record contained in the file. These attributes use the $FILE_NAME attribute to access attributes such as the file's name, size, modification date, creation date, and so on. By using the $DATA attribute, it is also found in which clusters this data is stored. Then, the data of the data size is obtained by going to the related folder and written to a file to save this file in a different area.

3. Results

In order to observe the results of this work, a computer program is coded and the data deleted from a storage device formatted as NTFS using the MFT file is recovered (Figure 3.1 and Figure 3.2). At the same time, the storage device was examined with a hex editor program.

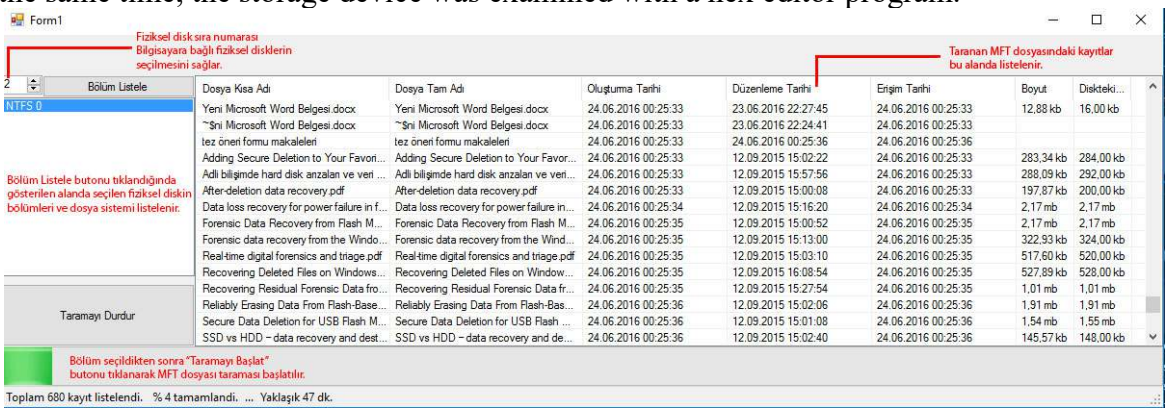

Figure 3.1 Data Recovery Application

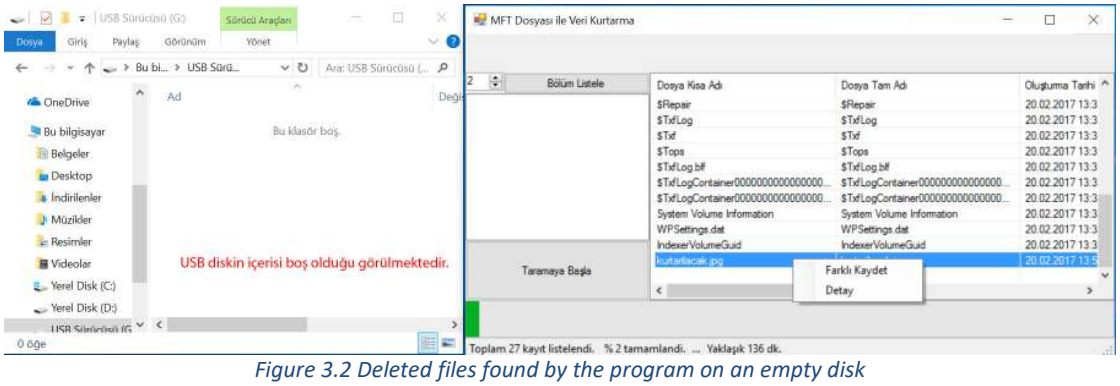

Figure 3.2 Deleted files found by the program on an empty disk

If the file size is large, the $DATA attribute parts of the file can be found in many different places. Data recovery will fail if some of these parts in different locations are corrupted.

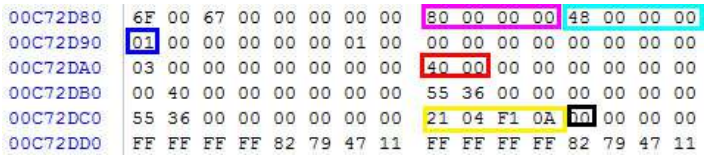

Figure 3.3 File part in different location (0x0AF1)

When a fully formatted disk is scanned, only metadata files are listed. In the quick formatting process, the files before formatting are also visible. If full formatting is done, the disk is completely reset and the MFT records are deleted. For this reason, recovery cannot be done with this method. After the files on the disk are deleted, another file is copied to the disk and the deleted file is displayed in the program list instead of the first one. Therefore, it is possible to recover all the files if a data recovery operation is performed after the batch erase operation or after the quick format operation without performing any writing on the disk.

4. Discussion

While the structure of the NTFS file system is found in the literature, there is no comprehensive study to recover deleted data. With this study, the academic literature has gained a comprehensive study on data recovery from NTFS file system using MFT file.
The data on the storage device can also be stored in a layered manner. This work is an easy and quick solution of data recovery methods. More extensive data recovery methods need to be improved in the literature.

Conclusions

This study aims to be an introduction to the researchers in this field. Increasing the importance of forensic evidence and the relief of forensic cases through the use of these data is crucial. Therefore, academic study in this area needs to be increased.

References

[1] Merrick, J. (2012, December 24). An Introductory Guide to Data Recovery. Retrieved October 03, 2018, from https://computers.tutsplus.com/tutorials/an-introductory-guide-to-data-recovery--mac-44549

[2] Naiqi, L., Zhongshan, W., Yujie, H. and Ke, Q. (2008). Computer forensics research and implementation based on NTFS file system. Proceedings - ISECS International Colloquium on Computing, Communication, Control, and Management, CCCM 2008, 1, 519–523. doi:10.1109/CCCM.2008.236

[3] Mahant, S. H. and Meshram, B. B. (2012). NTFS Deleted Files Recovery: Forensics View. IRACST -International Journal of Computer Science and Information Technology & Security, 2(3), 491–497.

[4] Ravindra, P., Kalal, R., Soumya and Mandal, V. (2013). Logical data recovery technique for USB devices. Proceedings - 2013 International Conference on Emerging Trends in Communication, Control, Signal Processing and Computing Applications, IEEE-C2SPCA 2013, 3–8. doi:10.1109/C2SPCA.2013.6749447

[5] Default cluster size for NTFS, FAT, and exFAT. (n.d.). Retrieved March 24, 2018, from https://support.microsoft.com/en-us/help/140365/default-cluster-size-for-ntfs-fat-and-exfat

[6] How NTFS Works: Local File Systems. (n.d.). Retrieved March 24, 2018, from https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc781134(v=ws.10).

[7] Carrier B. File system forensic analysis. Addison-Wesley; 2005. pp. 71-72.

[8] Disk Concepts and Troubleshooting. (n.d.). Retrieved March 24, 2018, from https://technet.microsoft.com/en-us/library/cc977221.aspx.

[9] Carrier B. File system forensic analysis. Addison-Wesley; 2005. pp. 201.

[10] Russon, R. Concept - Attribute Header. (n.d.). Retrieved March 24, 2018, from https://flatcap.org/linux-ntfs/ntfs/concepts/attribute_header.html.

[11] MBR and GPT Disks. (n.d.). Retrieved March 25, 2018, from http://www.cse.scu.edu/~tschwarz/coen252_07Fall/Lectures/NTFS.html.

[12] Russon, R. Concept - Data Runs. (n.d.). Retrieved March 25, 2018, from https://flatcap.org/linux-ntfs/ntfs/concepts/data_runs.html.

[13] Russon, R. NTFS - Attributes. (n.d.). Retrieved March 25, 2018, from https://flatcap.org/linux-ntfs/ntfs/attributes/index.html.