Multimedia Tools and Applications (2020) 79:16093–16111
https://doi.org/10.1007/s11042-019-7199-y

ExtSFR: scalable file recovery framework based on an Ext file system

Seokjun Lee[1] · Wooyeon Jo[2] · Soowoong Eo[2] · Taeshik Shon[2]

Received: 19 November 2018 / Revised: 29 December 2018 / Accepted: 9 January 2019 /
Published online: 29 January 2019

Abstract

As the technologies based on the Internet of Things, the Cloud, Big Data, and mobile technology have recently become the engine of the next-generation fusion environment, the use of consumer electronics with Linux/Unix-based operating systems which include mobile and embedded operating systems has been gradually increasing. As these technologies are applied in the real world, digital forensics and post-processing techniques in the next-generation environment are required for security/privacy perspective. In this paper, an Ext2/3/4 file system's file recovery framework which is suitable for the next-generation environment is proposed. Also, Ext4 used from small to large size file systems in recent consumer electronics such as home appliances, mobile devices, home-office devices, entertainment devices, etc. The proposed framework takes the various Ext4 file systems created in the IoT/Cloud/Big Data/mobile environment into account, and it is configured to accommodate not only Ext4 used mainly in the recent environment but also Ext2/3 legacy environment. Additionally, the proposed framework is implemented as a prototype and validated it by comparing it with the existing commercial technologies, showing that the accuracy and efficiency of the prototype of the proposed framework for large file system recovery rates are superior to those of the existing technologies.

Keywords Consumer electronics · Data recovery · Data security · Digital forensics · File systems

✉ Taeshik Shon
tsshon@ajou.ac.kr

Seokjun Lee
slee235@kennesaw.edu

Wooyeon Jo
dndusdndus12@gmail.com

Soowoong Eo
archinitus@gmail.com

[1] Department of Computer Science, Kennesaw State University, Marietta, GA 30067, USA

[2] Department of Computer Engineering, Ajou University, Suwon 16499, South Korea

1 Introduction

As consumer electronics become ubiquitous, these devices can have a various personal information, there is a potential privacy leakage threat. In order to prevent privacy leakage of the consumer electronics, it is necessary to consider not only the data present in the device but also the deleted file or the memory area which is deallocated. By studying the deleted data recovery technology, it is possible to improve the awareness of the personal information remaining in the unallocated area and to utilize those studies as a basis for preparing countermeasures against the privacy leak. On the other hand, consumer electronics are likely to be used when a crime is committed, and a clue or evidence can remain on those consumer electronics. In such circumstances, the demand for digital forensic techniques in a criminal investigation is increasing. The most common digital evidence includes photos, personal information, document files, and data sheets. This evidence can be recorded in physical memory as well as the file system. Though the technique to collect volatile information through analyzing physical memory is also being studied [6, 15], analyzing a file system is more effective. In a criminal investigation, deleted files should be recoverable and analyzable using standard techniques because all of the deleted files, which deleted by criminals, are evidence candidates. Therefore, in digital forensics, file system analysis, including the recovery of deleted files, is very important. In this paper, the words 'file recovery' and 'data recovery' are the core techniques for storage analysis of the digital forensic process, describing the recovered files that the suspect deleted deliberately from the file system.

In a criminal investigation, the investigation cases that involve analyzing personal computers, servers and other consumer electronics are increasing. According to a well-known international statistics companies, the trend over the most recent 3 yrs is that the Windows OS series continues to dominate the OS market used in desktops, mobiles, and tablets. However, Linux-based operating systems which are used to operate IoT devices, mobile devices, TV, and other consumer electronics, are steadily growing.

In the server market, Linux-based servers have been used more than Windows-based servers for a long time. This is especially true in the case of web servers where the use of Linux/Unix-based OSs is more than double that of the Windows OS shown in the 'Usage of operating systems for websites in 2018' from the one of the survey statistics.

Additionally, as the IoT, the Cloud, Big Data, and mobile industries are promoted, the use of consumer electronics with various Linux/Unix-based OSs, including Android and Embedded Linux, are expected to grow. Accordingly, the use of Ext/UFS file systems representing Linux/Unix will increase. Therefore, Windows forensic techniques have been studied to date, but the demand for forensic techniques in the field of Linux/Unix will increase in the near future. Specifically, any future technology development should consider the fact that the devices used in the fields of the Cloud and Big Data use various Ext4 file systems compared to the devices for personal users.

The techniques to recover a deleted file in a file system can be categorized into two groups. One uses the file system metadata, and the other uses file carving to recover the deleted file by anticipating its features. Though the metadata-based recovery technique may produce less quantitative recovery results than the file carving-based recovery technique, but it can detect a deleted file faster and more accurately. Therefore, it is advantageous to use metadata techniques to recover a file deleted to destroy evidence at the scene of a criminal investigation. This is true if the target of the investigation is a various Ext4 file system such as a server, where it is impossible to depend on the file-carving technique because of time constraints, technical

efficiency, and accuracy. Because most file-carving techniques recover meaningless binary data, analysis takes longer, and the accuracy of the recovered result has some problems. Additionally, the larger the file system is, the more the efficiency of recovery drops.

The contribution of this paper is to propose and verify a scalable framework that can analyze and recover a various Ext4 file system and be applied to a legacy Ext2/3 file system by considering the latest trends of the IoT, the Cloud, and Big Data industries. Additionally, proposed framework is implemented and verified it as a prototype and compared its results with the recovery results using commercial and public Ext file system file recovery tools. Currently, there are many analyses of consumer electronics such as mobile devices, smart home appliances, home-office devices, gaming consoles and general-purpose digital devices such as workstations, servers in various cases investigated by the police and prosecutor's office. As an expectation of the proposed file recovery framework in this paper will apply to the digital forensic process, especially the file system forensic process.

The remaining sections of this paper are as follows. The related works is introduced in Section 2 and explain the proposed framework and a structure that covers Ext 2/3/4 in Section 3. In Section 4, the description of the testing file system image of each Ext2/3/4 file system to verify the proposed technique and results alongside other recovery tool results is provided. There is some discussion in Section 5. Finally, the conclusions are presented in Section 6.

2 Related works

The techniques to recover deleted files in a file system are largely divided into two categories. One method is the technique of file-carving, which restructures and analyzes the deleted file based on the content of the file block, without any help from the file system information [12, 14]. The other method is to use the file system metadata, similar to the proposed framework in this paper.

File carving techniques have been studied for a long time. In the beginning, the main technique was to extract file data based on the value of the file signature at the start and end points of a file. More recently, advanced file-carving techniques using data-mining to complement the file signature method have been proposed in some research studies [18]. Studies targeting a file of a specific type, such as a multimedia file with a high recovery importance, have also been conducted in some papers [13, 19].

Because deleted file recovery techniques using file system metadata use the information dependent on a file system, a different technique is required depending on each file system. Additionally, many studies on forensic techniques based on metadata for NTFS (New Technology File System) and FAT (File Allocation Table) Windows-based file systems have been conducted in previous research studies [1, 8, 9, 20]. Alternatively, the XFS file system, used on UNIX file systems, with metadata-based deleted file recovery similar to our proposed scheme is described in a recent paper [11].

In the case of the Ext file system targeted in this paper, its analyzing method and recovery techniques for deleted files have been made public in various papers, educational materials, TR, and webpages. To understand the structure of the Ext file system, the disk layout of the Ext file system is provided in the Ext4 Wiki (https://ext4.wiki.kernel.org/) of the Linux Kernel Organization.

The recovery technique to find and recover a deleted file in a file system using metadata, and its basic analyzing technique, was published in 'File System Forensic Analysis (2005)' written by Brian Carrier [2]. A paper that improved upon the recovery technique for Ext2/3 based on Carrier's work has been published [7]. This paper improved the recovery technique based on the possible problems in Ext2/3 and conducted some comparisons for verifying its technique. In Ext3/4, unlike Ext2, some metadata is deleted when a file is deleted, so analyzing the journal area is required to recover it. This analysis technique was published in a technical paper on Ext3 Journaling, analyzing the method uploaded in the SANS Reading Room in 2007 [4]. However, the main purpose of the technique in this technical paper was to analyze backup data from the journal record, so it did not verify whether it could be applied to a file system image. In Ext4, a new file allocation scheme named 'extent tree' was introduced, and the analyzing technique for a deleted file was published in a paper on the analyzing method of a file system in Ext4 [5]. In this paper, techniques to analyze a file system were added as a module of 'The Sleuth Kit' [16], which is a well-known library and collection of command line tools to investigate disk images. However, it was not verified whether these techniques could recover a file deleted with a given theoretical process.

Additionally, technical documents and papers propose various tools for analyzing file systems supporting Ext2/3/4. Some of these support the Ext series while carrying out only file-carving regardless of the file system metadata. Some of the commercial recovery software and freeware support metadata analysis of Ext2/3/4. Because the major topic of the techniques published address 'how to find the deleted file', these techniques could not be verified as an automated tool or through comparison with other techniques. Considering the existing recovery techniques, they should be verified under the latest environment because Ext2/3/4 file systems in the Linux Kernel are also modified and updated continuously. Additionally, as the environmental factors have changed recently, techniques need to be compared and verified at the technical level between tools and deleted file recovery techniques when considering the large file size and various size of file system environment.

3 Scalable file recovery framework based on the Ext file system

3.1 ExtSFR framework: Ext scalable file recovery framework

The ExtSFR framework is divided into three phases as shown in Fig. 1. In the first phase, the entered file is analyzed and identified to determine what file it is, and the second phase is used to search for the deleted file. In the final phase, the file content blocks of the deleted file are collected, and the deleted file is recovered.

The mechanism of each part of the Ext2/3/4 is changed depending on which method is used between block mapping and extent. When designating the file content, it is necessary to consider whether a hash tree is used in the structure of the directory and whether the journal area is used when recovering a deleted file.

3.2 File system identification

To analyze a file system, you first need to identify the file system from the image file of the file system or storage media. If an image of the disk storage is acquired, the analyzer must reacquire the partitions using the partition entry information from the Master Boot Record

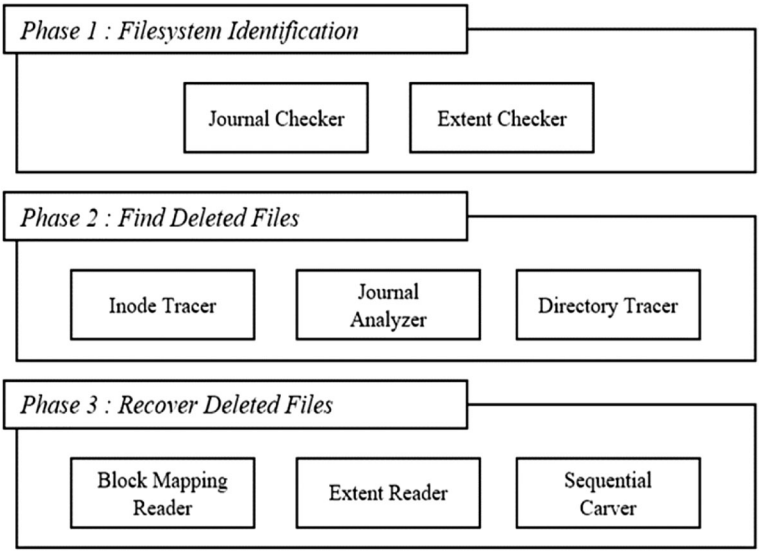


Fig. 1 Ext scalable file recovery framework

or GUID Partition Table which are located at the beginning of the disk image. If you have an image of a file system, you need to identify what kind of file system it is and what functions are set. Then, the appropriate analytical techniques to the file system should be applied.

The file system identification phase includes a pre-processing stage where the file system image is received and the file is processed. The most important work in this phase is the identification of the Ext2/3/4 file system. When the size of an image file is very large, a malfunction can occur in this phase, so it is important to prepare for handling this problem.

- *Module 1–1: Journal Checker*

To distinguish the Ext2/3/4 file systems, a method to check the default option of each file system is used. First, to distinguish the Ext2 from the Ext3/4 file system, we determine whether journaling is used. This is done by checking the on/off state of the flag 0×4 'Has a Journal' of the s_feature_compat field. This field is __le32 sized value and it is positioned at the offset 0x5C from superblock's starting address.

Though we can configure Ext2 to use a journal while mounting it on Ext3, using a journal means that the recovery tool should analyze the journal. In this case, even though the analysis of the file system may be inaccurate, there is no problem in analyzing the actual deleted file because the tool uses the same technique used in journal checking by recognizing the file system as Ext3.

- *Module 1–2: Extent Checker*

To distinguish Ext2/3 from Ext4, the analysis tool must check whether the target file system uses an extent structure or not. We can distinguish Ext2/3 from Ext4 by checking the on/off state of flag 0×40 'File in this file system uses extents' of the s_feature_incompat. Like journaling, Ext2/3 can also be mounted as Ext4 while activating several options. At this time,

though the determination of the original file system is impossible, there is no problem in analyzing the deleted file because its analyzing is processed the same way as Ext4.

There is a scalability issue in the File System Identification Phase. To analyze a large-size file system image file, the analysis tool must support 64-bit features. Generally, there are no problems in a small GB file, but it is impossible to read the whole range of a file with a 32-bit data type when the controlling files are dozens of GB to TB in size. Therefore, all the variables recording a position address in the source code or being used as file system pointers should use 64-bit operations. For example, in the C programming language, the long int type is used as a 64-bit integer in a 32-bit system (×86), but the long int type is used as a 64-bit integer in a 64-bit system (×64). Therefore, when these two types are used simultaneously, there is confusion due to the difference between the two systems (×86 and ×64). To solve this problem, we used the int64_t type that uses 64-bit data operations regardless of the platform. This type has the range of −9,223,372,036,854,775,808-9,223,372,036,854,775,807 (-2^{63}–$2^{63}-1$), so the file offset, having EB(Exabyte)-sized range, can be designated.

3.3 Find deleted files

Finding a deleted file based on metadata requires three steps. The first step is to find the inode of a deleted file by checking all of the inodes of the file system. The second step is to collect pairs of inode number-file names of a deleted file from the trace of the directory entry by checking all of the directories. The final step is to match the list of recoverable candidate inodes and the name of the deleted file by comparing the pairs. All three steps are processed in the Find Deleted Files Phase. However, an investigator with expert knowledge of file system forensics can use the inode for investigation. Therefore, if the inodes are located in the journal, it can be selected for recovery data.

- *Module 2–1: Inode Tracer*

When files are recorded in a file system, the value of the inode bitmap is set to 1, and when a file is deleted, the value is set to 0. Because it is natural that the value of the inode bitmap that was not used at the beginning is 0, we extracted all of the inodes of a deleted file except the initial one.

To do this, we first collected the inode numbers set to 0, considering only the items having values other than 0 at deletion time as a deleted file by checking the inode table for the inodes, adding it to the list, and managing it.

At this point, we must be careful with a very large file having an inode table and an inode bitmap placement at an offset larger than the 32-bit offset. The variable designating the offset of the file system image should be processed so that it can support 64-bit operations.

- *Module 2–2: Journal Analyzer*

When a file is deleted in Ext3/4, we cannot recover the file immediately because its size and content address values are initialized to 0, and the information must be found on a backup inode by analyzing the journal area [4]. The journal area is a temporary backup technology provided by the Ext3 and Ext4 files. Basically, this is a technique used to recover from file system corruption or operating system errors. However, when there is not enough deleted file information obtained by analyzing the inode, you can analyze the journal area and check

whether the information of the backed up inode exists. When we find an area that is presumed to be a backup inode, an inode whose file size field is not empty, the i_dtime field is filled with 0×00, and a value that remains in the file address field is presumed to be the backup of the inode; we recover the file based on this value. It is also preferable to check again by comparing the values that have particular characteristics of each file, such as i_generation in inode and backup inodes.

- *Module 2–3: Directory Tracer*

To identify the deleted files, directory entries of all files should be inspected by tracing all of the directories starting from the root directory. When the file is a 'regular file' designated by a directory entry from all the directories, and if an applicable inode number exists when compared with the list of deleted files, we extract the file name from the directory entry and record it. In Ext2/3, analyzing the list of directory entries is required. In Ext4, we must consider a newly added structure named 'Hash Tree' directories.

There is a scalability issue in the Find Deleted Files Phase. In a very large Ext4 file system, when many files are recorded in one directory, the directory is managed with the hash tree structure to enhance the indexing efficiency. To be compatible with Ext2/3, this structure uses a method to expand the structure of the directory entry by using the length field. Therefore, for the file allocated to the file system, the mechanism used for analyzing the directory entry in Ext2/3 can be applied as it is, but the structure of the hash tree must be analyzed for higher accuracy. Specifically, because we need to analyze the entry with a disconnected link to analyze the directory entry of a deleted file, the hash tree structure should be applied.

3.4 Recovery of deleted files

After searching for a deleted file, we must recover a file selected by a user or all of the recovery candidate files that were classified as recovery data. The method to secure file content from the inode of a deleted file is the same as the file read mechanism of Ext2/3/4. Therefore, the method for recovering a deleted file using the information in the file content addressing the deleted file inode is used, and the file content can be recovered successfully as long as the file content area is not overwritten. In Ext2/3, the traditional block addressing method should be used, while in Ext4, extent should be used.

- *Module 3–1: Block Mapping Reader*

File sizes using block mapping are shown in Table 1.

When using the triple indirect pointer in Ext2/3, the maximum size of a single file is calculated as 4 TB, meaning that a recovery tool should be able to provide a triple indirect

Table 1 A file size can be represented by each block pointer

Block Pointer	The Number of Blocks	File size (Block size = 4 KB)
Direct Pointer	12	48 KB
Indirect Pointer	1024	4 MB
Double Indirect Pointer	1024^2	4 GB
Triple Indirect Pointer	1024^3	4 TB

pointer to support very large files. As long as a file is not overlaid with other file data after the file content is deleted, files with sizes shown in the table above can still be recovered.

- *Module 3–2: Extent Reader*

Because the journal technique is used in Ext4, as in Ext3, a deleted file can be recovered if the backup inode data for the deleted file remains in the journal area. In a broad sense, the recovery method is the same as the one in Ext3; however, there is a difference in the recovery method due to the increased size of the inode structure in Ext4 (256 bytes vs. 128 bytes), and the file content addressing method uses the extents tree structure rather than block addressing as in Ext2/3 [5]. Table 2 shows the calculated maximum file sizes according to the depth of the tree when extent is used in Ext4. When the depth is 2, the theoretically recordable file size falls outside of the structural limit of the Ext4 file system. Therefore, because the depth is 2, it is possible to recover a very large file.

- *Module 3–3: Sequential Carver*

As the Linux Kernel versions and OS distributions are updated, a file recovery challenge occurs in the Ext2/3 file system.

When conducting file system imaging in an activated state without unmounting the file system, we discovered that imaging was performed such that the block in which the indirect address values were stored was filled with 0×00. To solve this problem, we proposed a sequential carving method [7].

The Ext file system has a characteristic where it tries to place file blocks as closely as possible when the file size is relatively small. Taking advantage of this characteristic, we can estimate the locations of the file blocks from the starting point, through which we can recover a deleted file without the values of the indirect blocks.

Table 3 shows the estimated maximum sizes of files that can be recovered with the sequential carving method. Because sequential carving succeeds only when a file is stored sequentially, the minimum condition is that a file size is smaller than the size of the block group, the size of metadata block.

4 Experiments

We generated a file system with a 1 GB general partition and a 1 TB partition with formed file datasets, to verify the prototype tool developed based on ExtSFR proposed in this paper, for each case, Ext2/3/4. To accomplish this, we created a target file system using a combination of file addition and deletion. To minimize the dependency on the system, it was developed using the C language based on an Ubuntu Linux 14.04 LTS environment. Additionally, because the

Table 2 A file size that can represent each block pointer in Ext4

Block Pointer	The Number of Blocks	File size (Block size = 4 KB)
Extent tree (depth = 0)	$4*2^{15}$	512 MiB
Extent tree (depth = 1)	$4*340*2^{15}$	170 GiB
Extent tree (depth = 2)	$4*340^2*2^{15}$	56 TiB (Max 16 TiB)

Table 3 Estimated file size can be recovered by sequential carving

Block Size	Block Group Size	Estimated File Size Can Be Recovered by Sequential Carving
1024 bytes	< 8 MiB	< (8 MiB-Metadata Blocks)
2048 bytes	< 32 MiB	< (32 MiB-Metadata Blocks)
4096 bytes	< 128 MiB	< (128 MiB-Metadata Blocks)

target system using the Ext file system is a mostly Linux-based environment, it can be applied to situations where a recovery tool must be executed in a running target system. The prototype was tested on versions from Ubuntu 12.04 LTS and Ubuntu 16.04 LTS.

4.1 Test environment

The target file system was generated in an Ubuntu Linux 14.04 LTS environment with a 1 GB partition derived file system image using the Linux dd command and a 1 TB partition derived image using the one of major commercial disk imaging hardware as depicted in Fig. 2. With dedicated disk imaging hardware that includes write-protection technology, reliable disk and file system images can be obtained without damage to the original disk. The major features of disk imaging hardware which is used for the experiments are shown in Table 4.

4.2 Test dataset

Using the target file system generated by an Ubuntu Linux 14.04 LTS environment and target file system sizes of 1 GB and 1 TB, we created various file types, including jpg, wmv, and pdf, with several file sizes ranging from KBs to several MBs to build the file dataset for the experiment. Because the proposed framework operates based on metadata, the file type is irrelevant to the recovery performance. However, the recovery mechanism of the existing commercial tools is not available, so we configured various file types to compare the performance. Also, we can evaluate the other tools using file carving techniques for the specific file types.

Fig. 2 File system imaging using commercial imaging hardware

Table 4 Forensic dossier features

Feature	Description
Capture Speed	Capture speeds over 7 GB/min
Captured Image Format	DD image (full disk) E01 file format supports Encase® v6.x and FTK® v3.x and above.
Hardware Interfaces	SATA, IDE, Firewire, USB, SCSI, SAS
Authentication	MD5 and SHA-256 authentication
Write Protection	Write-protected Data Transfer to Prevent Overwriting (Built-in)
Erase Options	Standard Wiping, High-speed Security Erase

- *Dataset for 1 GB Partition*

We generated 30 files that had different file sizes from several KBs to hundreds of MBs and used them as file sets. Table 5 shows the basic file dataset for the experiments used. Additionally, the MD5 hash values for each file were recorded to check whether the resulting file has been completely restored after the file recovery is performed.

- *Dataset for 1 TB Partition*

After generating file sizes from several KBs to dozens of MBs to be used as datasets, we added a 110 GB (very large) file. In file systems of 1 TB or more, the default block size is set to 4 KB, and when the triple indirect pointer is activated in Ext2/3, single files greater than 4 GB can be managed. Even in Ext4, when the extent depth is greater than 1 GB, files can be managed.

4.3 Comparison with other software

To verify our proposed method, we compared the recovery results of the ExtSFR prototype (Fig. 3) with those of file recovery software, one of the commercial comparison target software as a 'Product A' [17], and one of the freeware as a 'Product B' [3] (Fig. 4).

'Product A' is one of the most well-known file system recovery software that supports Ext2/3/4 with several file system-related features, including immediate data access, recover lost partitions, reconstruct RAID, deleted files recovery, etc. However, on the website, they mention that 'the quality of data recovery result may be low due to the file system design' on the Ext2/3/4 file system.

Table 5 Example of the basic file dataset for experiments

	Filename	Size	Hash (MD5)
1	Image(o1 MB) (0)_A.jpg	3.5 MB	f151234431f74f4154b1933427a5e711
2	Image(o1 MB) (4)_A.jpg	1.3 MB	8991311d7fcab23a0a7431dd2e9ebad2
3	Image(u100 KB) (1)_A.jpg	64 KB	66dfc09e4b030fca63be39427ba38b9e
4	Image(o100 KB) (3)_A.jpg	332 KB	91ad98b178c9729b4fc9152e27034dd8
5	2min_winvideo_A.wmv	46 MB	b8b1edabe2710fd5a559fd7675f6460c
6	5min_winvideo_A.wmv	116 MB	5ccd8b20b36be81b74dd6536566ebf46
7	Hancom_doc_3_A.hwp	52.6 MB	3b91d538a71ff27f502ad640cfc534bb
8	Hancom_pdf_1_A.pdf	143 KB	197db34099bdb4e9133e429673b3095e
9	MS_Excel_2_A.xlsx	464 KB	4fbfacb07f0912eb5e071cd3c9716bff
10	MS_Slide_3_A.pptx	4.8 MB	05e902ef71a2f775a75e3a7dd88e259d

```
base address = 0
File Read Completed.

FILESYSTEM is EXT4


**DEBUGING USING ONE FILE - ver.0.9.0 Directory Entry Tracing**

=========================================================
Analysis for Ext4 function Start
=========================================================
1. Superblock analyzing - Complete

2. Group Descriptors analyzing - Complete

3. inode Tracing - Start

-----------------------------------------------------------
                In Ext 4 inode_tracer
-----------------------------------------------------------
  The total number of inodes            = 63060(hex)   , 405600(dec)
  The total number of Block groups              = 32(hex)   , 50(dec)
  The number of inodes per Block group (inodes/block group)     = 1FB0(hex)   , 8112(dec)
  Block size                    = 1000(hex)


 - The number of inactive inodes = 308914

 - Deleted inodes / unused inodes / Journal - Analyzing start

        >> The number of confiremd inodes (inactive inodes) : 308913
        >> The number of deleted files : 44
        >> Current inode : 405600, Current Block Group : 50

 -------------------------------------------------------------
 (Result) The number of deleted files : 44
 -------------------------------------------------------------
 - Ext4 inode Tracer - Complete
 - inode Tracing - Complete

4. Directory Entry Tracing - Start

 - Directory Entry Tracing - Complete

=========================================================
Recovery Process
=========================================================
Recovering... ( File size : 3372 )
A File Recovery Complete. ( File name : 1767 )
Recovering... ( File size : 320A )
A File Recovery Complete. ( File name : 1767 )
Recovering... ( File size : 3993 )
A File Recovery Complete. ( File name : 1767 )
Recovering... ( File size : 3295 )
```

Fig. 3 ExtSFR prototype screenshot (Ext4 Image Used)

To compare the file recovery results, we recorded the MD5 hash values of the files before deleting and calculated the MD5 hash values of the recovered files, and then we compared not only the recovery rates of file content but also the file names.

In this paper, we compared our prototype based on the proposed framework with the existing software. Additionally, its performance was evaluated through comparison of hash values after recovering a deleted file. According to these comparisons, the prototype using the proposed framework showed superior performance. The results are shown in Figs. 5, 6, and 7. All of the results are the same format, with the number of 'Number of Deleted Files' in the first column, 'Recovered Files' in the second, 'MD5 Value Matched' in the third, and 'Recovered File Name' in the fourth. If one file is fully recovered, the number of all areas will be

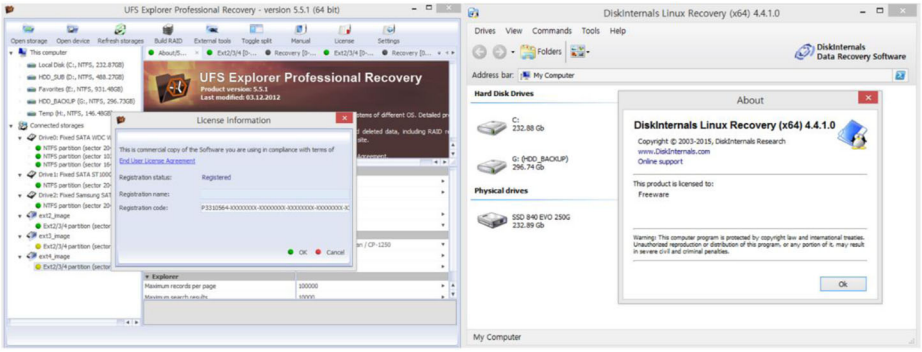


Fig. 4 Comparison Target 'Product A' and 'Product B'

incremented by 1. If the file name does not match but the contents are completely recovered, only the 'Recovered Files' and 'MD5 Value Matched' was incremented.

Figure 5 shows the comparison results on an Ext2 1 GB file system image. The 'Recovered Files' in the Fig. 5 means a number of recovered files whether each file has a data or not. These could be well-recovered files or wrong-recovered files. To check the definite recovery results, we also compared the MD5 value of the recovered files with the MD 5 of the deleted files, which were calculated before being deleted. The value named 'Same MD5 Value' shows the most important data in the comparison result. The 'Product B' recovered 60 files from the target image, and only 3 files of them matched the MD5 value from the original files. This means that 'Product B' recovers deleted files using a file carving solution, and it creates too many incomplete file chunks. 'Product A' recovered 21 files, and only 3 of them have the same MD5 value as the original files. In contrast, the proposed framework recovered 30 files, and 22 of them matched the MD5 value from the original files. Additionally, the file names also recovered more often than the other tools. Therefore, the other two tools do not consider metadata in an Ext2 file system as much as our proposed framework.

Figure 6 shows the comparison results on the Ext3 1 GB file system image. 'Product B' recovered 86 files, though the image had only 30 deleted files. Additionally, this tool only recovered 5 files having the same MD5 value as the original files. It is hard to use in a real-world data forensics situation because it creates too many meaningless files. On the other hand, 'Product A' recovered more files than the proposed framework, but our framework recovered

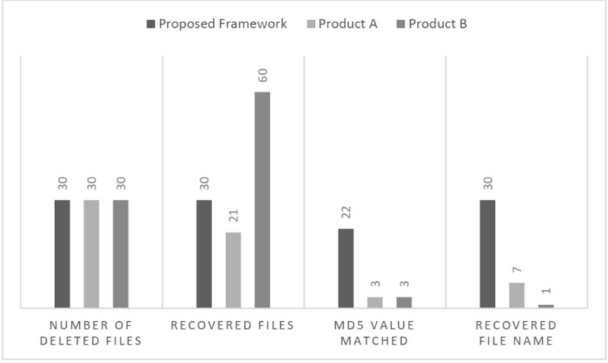


Fig. 5 Comparison Results on 1 GB Filesystem (Ext2)

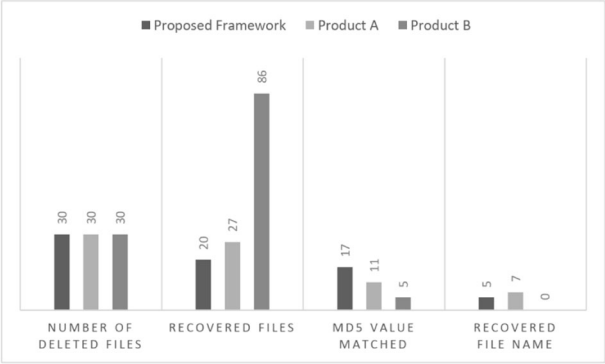


Fig. 6 Comparison Results on 1 GB Filesystem (Ext3)

more files which have the same MD5 value as the original files. However, 'Product A' is better for finding file names than our framework. 'Product B' does not work on the file name recovery.

Figure 7 shows the comparison results on an Ext4 1 GB file system image. In this Ext4 comparison result, our proposed framework is slightly better than other tools. Our framework recovered all files with 15 filenames. However, other tools were also successful at file recovery even though these tools could not recover the file names. It appears that these tools handle the Extent structure which is the default used in the Ext4 file structure.

However, we cannot explain what occurred because other tools do not provide their recovery mechanism. These experimental results show that our proposed framework is more efficient than the two other tools.

Additionally, for a very large file system image, we constructed a 1 TB file system with a dataset in each of the Ext2/3/4, and additionally, a 110 GB file was added to test recovery of a single very large file. With these test file system images, we verified that our proposed framework can recover a single file over one hundred gigabytes. When comparing the results using the 1 TB file system, it was impossible to determine the file recovery results because 'Product B' carved numerous file chunks through their file carving technique. For this reason, we excluded it from the comparison results. When we compared the results on the 110 GB file, the existing tools could not recover the file normally, whereas the proposed prototype

Fig. 7 Comparison Results on 1 GB Filesystem (Ext4)

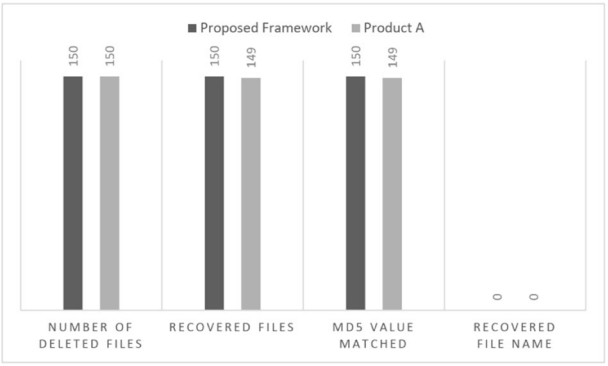


Fig. 8 Comparison Results on 1 TB Filesystem (Ext2)

successfully processed and recovered it accurately. The comparison of the results using the 1 TB file system is shown in Figs. 8, 9, and 10.

Figure 8 shows the comparison results on the Ext2 1 TB file system image. In this case, the proposed framework recovered all of the files but could not recover file names at all. 'Product A' also could not recover file names, and it could not recover a single large-size file. This result shows that 'Product A' cannot recover a deleted file with a deep depth structure. Additionally, the reason the tools could not recover file names is as follows. When we deleted files from our target file system, the links connecting the directories and files were disconnected. Thus, there is no evidence in the file system image regarding the link to deleted files and literal strings representing a file name.

Figure 9 shows the comparison results on an Ext3 1 TB file system image. The proposed framework recovered files that have the same MD5 value as the original files better than the 'Product A'. However, the result of file name recovery of 'Product A' is still better than our framework. Our framework is designed for checking every deleted file in an inode table and restoring the deleted inode data from the journal. With this result, it is can be assumed that the 'Product A' has an advanced feature that is used to retrieve file names. For instance, one of the methods to get file names is searching the strings, which are shaped similarly to the file names from the journal area.

Figure 10 shows the comparison results on the Ext4 1 TB file system image. On the 1 TB Ext4 file system image experiment, the proposed framework recovered more files and file names compared to 'Product A'.

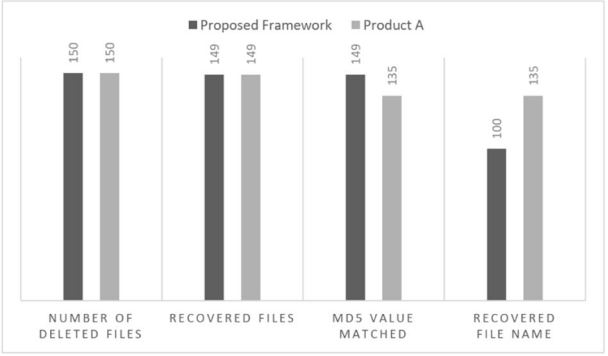


Fig. 9 Comparison Results on 1 TB Filesystem (Ext3)

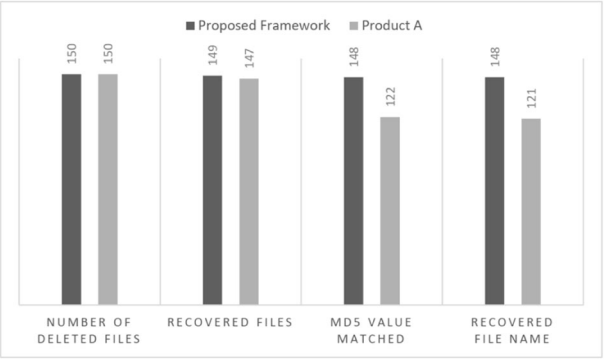


Fig. 10 Comparison Results on 1 TB Filesystem (Ext4)

The overall comparison results with the file recovery rates are shown in Fig. 11. This is a graph showing the recovery rates of all deleted files from the two images (1 GB, 1 TB) shown in the above experiment for three file systems (Ext2/3/4) and three tools including our proposed framework. The types of test datasets are expressed as color clarity for each tool and are listed in the order of the file system. We used MD5 hash values to verify the recovery rate of the file data, which is the most meaningful factor in digital forensics. For the test file system with the file dataset, the proposed framework showed performance superior to the existing tools with respect to the recovery rate. Regarding digital forensics, there may be a deviation in the analysis results depending on the state of the analysis target. Therefore, this experimental result does not represent an absolute recovery rate. This paper presents a framework for the Ext file system, considering the appropriate technology for each file system in each phase, showing a better recovery rate from a properly constructed test file system image.

Regarding processing speed, when analyzing 1 TB without the file carving option, 'Product A' spent approximately 50 min, whereas the proposed framework spent approximately 25 min.

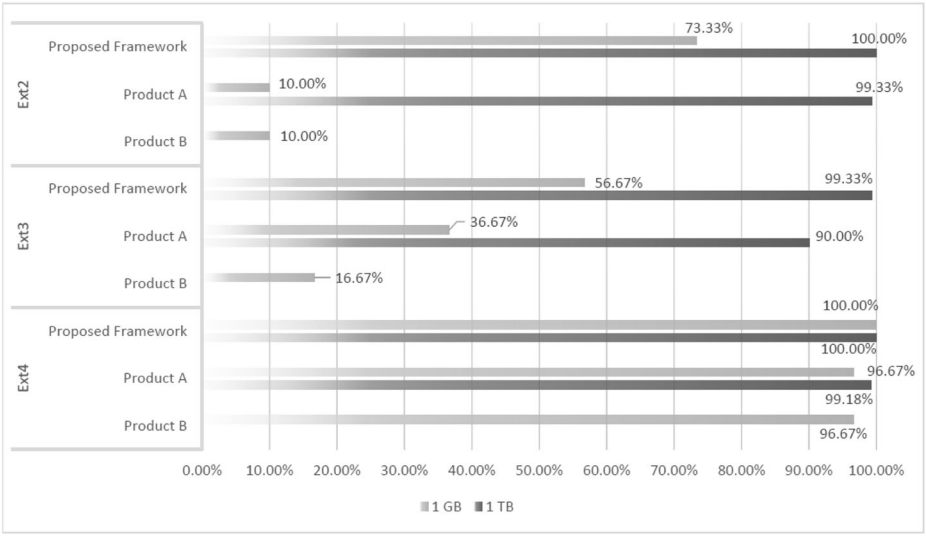


Fig. 11 Comparison Results on 1 TB Filesystem (Ext4)

Additionally, using only the modules of the proposed framework, we decreased the processing time by approximately 50% and obtained better recovery performance when compared to the commercial tool 'Product A'. Additionally, for both the very large partition and very large file, all of the Ext2/3/4 file systems recovered data faster and more accurately.

5 Discussion

In this paper, a prototype of the Ext file system recovery tool implemented on the basis of the proposed framework was compared with other tools. We validated the structure of the proposed framework. In the experimental results in Section 4, considering 'Product B', where file carving is presumed to be the main function, the larger the size of the analysis target file system image is, the more junk files were created. This is difficult to use in a digital investigation scene because it has to analyze a large number of meaningless files. In 'Product A', some experiments recovered more file names than the proposed framework, but the deleted files were recovered less often than the proposed framework. Regarding digital forensics, it is considered more important to recover the actual contents of the file than to recover the file name. Therefore, better file recovery is performed by applying the proposed framework.

We should also be aware of 'unintended' anti-forensics. There is a possibility that the implementation of digital forensics is blocked because of the TRIM technology. However, the experiments in the study show that the Ext4 file system has a very high recovery rate under high usage conditions, unlike other file systems (NTFS, HFS +) that support TRIM [10]. Fortunately, we are targeting very large file systems and Ext series file systems, so this is not yet a problem. However, if SSD is introduced into the server devices according to future technology development, it may become a serious problem. Therefore, additional experiments will be required in a variety of environments with a difference in the OS or hard drive type.

6 Conclusion

In this paper, we proposed a framework for recovering a deleted file in an Ext2/3/4 file system that can be applied to the next-generation environment and verified that it is more effective than the existing commercial tools through comparison data. The proposed framework could be applied not only to the various Ext4 file systems generated by the IoT, the Cloud, Big Data, and mobile industries in the next generation environment but also to the Ext2/3/4 environment in a legacy environment. Though we could not compare the specific recovery technologies of the various commercial and freeware tools because the internal structures are proprietary, we did find superior overall performance using the proposed framework. For each Ext2/3/4, we made samples of a 1 GB file system and a 1 TB file system and compared their recovery results, which showed that the proposed framework has superior performance over the others. Additionally, for 100 GB or greater sized files, we verified that the proposed framework could process them properly. Our main contributions are to provide a framework that can support large deleted files, support various size of Ext4 file system image files and have compatibility with Ext2/3 file systems.

Future studies based on this paper will accommodate various file systems by expanding the frameworks and adding the modules required to analyze each file system. The additional modules must be capable of being reused in each of the three phases of file recovery, that is,

file system identification, analyzing deleted files, and file recovery, by extracting the parts separately that can be applied to various file systems. For the file system framework to be used in criminal investigation more effectively, we need further studies on methods to apply file recovery techniques while minimizing their effect on the running target.

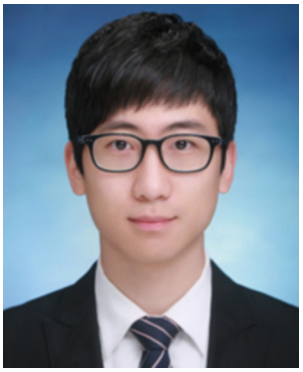
Acknowledgment This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No.2018-0-01000, Development of Digital Forensic Integration Platform).

References

1. Alazab M, Venkatraman S, Watters P (2009) Effective digital forensic analysis of the NTFS disk image. Ubiquitous Comput Commun J 4(1):551–558
2. Carrier B (2005) Filesystem forensic analysis. Addison-Wesley
3. Diskinternals Linux Recovery, http://www.diskinternals.com/, Accessed Sep 1, 2018
4. Gregorio N (2007) Taking advantage of Ext3 journaling filesystem in a forensic investigation. SANS Institute:10–35
5. Kevin D (2012) An analysis of Ext4 for digital forensics. Digit Investig 9:S118–S130
6. Lee S, Shon T (2014) Physical memory collection and analysis in SmartGrid embedded platform. MONET, ACM/Springer 19(3):382–391
7. Lee S, Shon T (2014) Improved deleted file recovery technique for Ext2/3 filesystem. J Supercomput 70(1): 20–30
8. Nabity P, Landry BJL (2010) Recovering Deleted and Wiped Files: A Digital Forensic Comparison of FAT32 and NTFS File Systems using Evidence Eliminator, Proceedings of the Southwest Decision Sciences Institute (SWDSI)
9. Naiqi L, Zhongshan W, Yujie H (2008) QinKe, computer forensics research and implementation based on NTFS file system, computing, communication, control, and management, 2008. CCCM '08. ISECS Int Colloquium 1:519–523
10. Nisbet A, Lawrence S, Ruff M (2013) A forensic analysis and comparison of solid state drive data retention with Trim enabled file systems, 11th Australian digital forensics conference. Edith Cowan Univ:103–111
11. Park Y, Chang H, Shon T (2015) Data investigation based on XFS filesystem metadata. Multimed Tools Appl 75(22):14721–14743
12. Piper S, Davis M, Manes G, Shenoi S (2005) Detecting hidden data in Ext2/Ext3 filesystems, advances in digital forensics. Int Fed Inform Proc 14:245–256
13. Poisel R, Tjoa S, Tavolato P (2011) Advanced File Carving Approaches for Multimedia Files, Journal of Wireless Mobile Networks, Ubiquitous Computing and Dependable Applications. 42–58
14. G.G. Richard, V. Roussev, Next-generation digital forensics, Commun ACM 49(2), 2006, pp. 76–80
15. Seo J, Lee S, Shon T (2015) A study on memory dump analysis based on digital forensic tools. Peer-to-Peer Network Appl Springer 8(4):694–703
16. The Sleuth Kit, http://www.sleuthkit.org/sleuthkit/, Accessed Sep 1, 2018
17. UFS Explorer Professional, http://www.ufsexplorer.com/, Accessed Sep 1, 2018
18. Veenman C (2007) Statistical disk cluster classification for file carving. Proc IAS '07 Proc Third Int Symp Inform Assur Sec:393–398
19. Yoo B, Park J, Lim S, Bang J, Lee S (2012) A study on multimedia file carving method. Multimed Tools Appl 61(1):243–261
20. Zhao S, Fei J, Liu N, Wu D (2008) Research and implementation of data recovery on windows NTFS. Comput Eng Des

SEOKJUN LEE received his Ph.D. degree in Computer Engineering from Ajou University, Suwon, Korea in 2017 and his B.S. degree in Computer Engineering from Ajou University, Suwon, Korea in 2011. He was a visiting research scholar in the Department of Computer Science, University of Georgia, Athens, GA, USA, in 2017. This author became a Member (M) of IEEE in 2017. Currently, he is a postdoctoral researcher of Department of Computer Science at Kennesaw State University, Marietta, GA, USA, in 2018. His research interests include Digital Forensics, Anomaly Detection, Network Security, Industrial Control System Security. Dr. Lee was awarded the Best Paper Award from the Korea Information Processing Society Conference in 2012, the Special Award from National Cryptography Contest from Korea Institute of Information Security in 2014.

WOOYEON JO received the B.S. degree in computer engineering from Ajou University, Suwon, Korea, in 2015. This author became a Member (M) of IEEE in 2017. Since 2015, he is in the M.S./Ph.D. integrated program, Ajou university, Suwon, Korea. His research interests include the development of digital forensic tools and techniques that utilize metadata from various filesystems, network forensics, extension to digital forensic science, applying digital forensics on control systems.



SOOWOONG EO received his M.S. degree in Computer Engineering from Ajou University in 2016 and B.S. degree in Computer Engineering from Ajou University, Suwon, Korea, in 2014. Currently, he is in Brauther Games, Korea. His research interests include the development of digital forensic software and forensic techniques about various types of filesystem, digital forensic science, applying digital forensics on control systems.

TASHIK SHON received his Ph.D. degree in Information Security from Korea University, Seoul, Korea in 2005 and his M.S. and B.S. degree in Computer Engineering from Ajou University, Suwon, Korea in 2000 and 2002, respectively. While he was working toward his Ph.D. degree, he was awarded a KOSEF scholarship to be a research scholar in the Digital Technology Center, University of Minnesota, Minneapolis, USA, from February 2004 to February 2005. From Aug. 2005 to Feb. 2011, Dr. Shon had been a senior engineer in the Convergence S/W Lab, DMC R&D Center of Samsung Electronics Co., Ltd. This author became a Member (M) of IEEE in 2014. Currently, he is a visiting professor of Electrical Computer Engineering Department at Illinois Institute of Technology, Chicago, USA, in 2017. He is currently a professor at the Department of Cyber Security, College of Information Technology, Ajou University, Suwon, Korea. His research interests include Industrial Control System, Anomaly Detection Algorithms, and Digital Forensics. Dr. Shon was awarded the Gold Prize for the Sixth Information Security Best Paper Award from the Korea Information Security Agency in 2003, the Honorable Prize for the 24th Student Best Paper Award from Microsoft-KISS, 2005, the Bronze Prize for the Samsung Best Paper Award, 2006, the Second Level of TRIZ Specialist certification in compliance with the International TRIZ Association requirements, 2008, and the Silver, Bronze, Excellent Publication Prize for Ajou University Award, 2013, 2014, 2016. He is also serving as a guest editor, an editorial staff and review committee of Computers and Electrical Engineering - Elsevier, Mobile Network & Applications - Springer, Security and Communication Networks - Wiley InterScience, Wireless Personal Communications - Springer, Journal of The Korea Institute of Information Security and Cryptology, IAENG International Journal of Computer Science, and other journals.