Forensic Analysis of Android Phone Using Ext4 File System Journal Log

Dohyun Kim, Jungheum Park, Keun-gi Lee and Sangjin Lee

Abstract As announcing Android OS 2.3, Gingerbread, Google changed the existing file system, yaffs2 to ext2 and adopted it as official file system in android phone. Ext4, the most widely used file system in Linux, not only assists large, but also provides fault tolerance through journaling function by adopting JFS—journal file system. In journal log created through journaling function of ext4, every transaction occurred in file system is record. All transactions include all events (e.g., creating, deleting, and modifying). Therefore, analyzing journal log, we would know what file did android user access to; could recover deleted files as finding the information of previous status of them. Moreover, we could also analyze user actions if we make up timeline by utilizing timestamp recorded in journal log. Based on these facts, in this paper, we aim to analyze journal log area in ext4 file system; to develop the tool, JDForensic, that extracts journal log data to recover deleted data and analyze user actions. This tool will be usefully utilized in the first time digital forensic investigation of android phone.

Keywords Digital forensics · Android phone · Ext4 file system · Journal log · Data recovery · Analysis of user actions

D. Kim · J. Park · K. Lee · S. Lee (✉)
Center for Information Security Technologies, Korea University,
Seoul, South Korea
e-mail: sangjin@korea.ac.kr

D. Kim
e-mail: exdus84@korea.ac.kr

J. Park
e-mail: junghmi@korea.ac.kr

K. Lee
e-mail: lifetop@korea.ac.kr

James J. (Jong Hyuk) Park et al. (eds.), *Future Information Technology, Application, and Service*,
Lecture Notes in Electrical Engineering 164, DOI: 10.1007/978-94-007-4516-2_44,

1 Introduction

For the last 2 years, android phone, the first smart phone launched by Google, has been watched by all around the world with interest, and has taken up the greatest market share in worldwide smart phone market.

At the first time, Google accepted yaffs2 as official file system in android phone. Unfortunately, yaffs2 had several problems with system stability and I/O speed. Also, it is not enough to support large file and not suited for multi-core device. In order to find out improved file system that makes up for the drawback of yaffs2, many of android phone companies such as Samsung, LG, HTC and Motorola have devoted themselves to find out suitable file system (e.g., Samsung: rfs, LG: ext3, Google/Motorola/HTC: yaffs2) for android phone equipped with large memory and multi-core device. Rfs provides high stable process but its I/O speed is too slow, ext3 processing is fast but not enough to support large file. For these reasons, Google introduced gingerbread (android OS version 2.3) and officially changed its file system to ext4 that compensates these several defects.

Ext4, an upgraded file system of ext3, is the system that adopts JFS—Journaling File System; the most often used in Linux system. This future-oriented file system—ext4, supports large file and suited for multi-core device as well as provides high I/O speed. Simultaneously, by strengthening journaling function, it helps not only swift and stable recovery but also prevents from unexpected event of system crash or power failure. In other words, ext4 provides higher levels of fault tolerance.

In ext4, there is journal area where numerous journal logs created by journaling function. This journal logs are created whenever events, namely transaction comes about. Before transaction is committed to file system in journal log, this journaling function will record the present status of metadata such as a directory or file. The purpose of this recording is for revoke when transaction committing is not successful. If committing fails, ext4 could have revoke done based on metadata in journal log. Thus, the data stored in journal log is the most important information for file system fault tolerance [1]. Furthermore, since containing metadata which all of changed data exist, this journal log is very much important to conduct data recovery as well as analyze user actions in the viewpoint of digital forensic investigation.

On account that ext4 has special feature that deletes file along with data block pointer at extents structure in inode table, we only use carving as method of file recovery at ext4 file system. Unfortunately, there is disadvantage when we perform carving which targets all unallocated area. It takes much time to perform it and has low accuracy on recovery. If finding previous metadata at journal log before conducting carving, we could obtain some information such as the name of file, is amount and timestamp. Besides, we could recover deleted data faster and more accurate than carving [2].

Descriptor block in the first block of journal log has a sequence number of the transaction. Furthermore, at the commit block or revoke block in the last block of

journal log, there is timestamp which commit or revoke occurred. Consequently, with sequence number and timestamps, we can make up timeline and then analyze user actions.

In this paper, we analyze journal log where the ext4 file system exists and recover deleted files. After checking out timestamp left in journal log, we also make up timeline and take user actions out of it. Moreover, we develop useful tool, JDForensic, which automate this process so as to be used at the first stage of digital forensic investigation about android phone.

2 Related Work

2.1 The Way to Extract Android Phone Data Partition

Before starting to analyze internal file system of android phone, we need image file of data partition which will be analyzed. There are two ways of Imaging—physical imaging and logical imaging. Physical imaging is carried by JTAG while logical imaging is carried out by "DD" instruction after rooting is executed.

In this paper, we will limit ourselves to the consideration of logical imaging file. To carry out logical imaging, applicable android phone must be rooted beforehand. Rooting of android phone is able to be conducted in the way of exchanging kernel in system partition. It means that we exchange existing kernel to another kernel where "SU" instruction inserted. The way of exchange kernel is conducted throughout recovery mode booting which autonomously supports android phone itself and this process affects only the system partition [3]. As a result, the integrity of data partition, the research subject of forensic investigation, can be retained.

We experiment with two devices—Samsung Galaxy S II and Google Nexus S. The imaging of two sets of devices are carried out after rooting is conducted. Imaging data partition by using "DD" instruction in Fig. 1.

2.2 Ext4 File System

Ext4, the most widely used file system in Linux, is the system for making up for ext3 in terms of not only assisting large file, but also providing fault tolerance through journaling function by adopting JFS-journaling file system. As the successor of ext3, it also has great advantage of simple upgrading ext2, ext3 file to ext4.

Ext4 file system is composed of large number of data unit which called block and these blocks are managed in several block groups. Every block group is in equal size; the numbers of block groups are variable depending on the size of disk. See Fig. 2 for layout of ext4 file system. Each block group has super block,

```
# mount
mount
rootfs / rootfs ro,noatime,nodiratime 0 0
tmpfs /dev tmpfs rw,relatime,mode=755 0 0
devpts /dev/pts devpts rw,relatime,mode=600 0 0
proc /proc proc rw,relatime 0 0
sysfs /sys sysfs rw,relatime 0 0
none /acct cgroup rw,relatime,cpuacct 0 0
tmpfs /mnt/asec tmpfs rw,relatime,mode=755,gid=1000 0 0
tmpfs /mnt/obb tmpfs rw,relatime,mode=755,gid=1000 0 0
tmpfs /mnt/usb tmpfs rw,relatime,mode=755,gid=1000 0 0
tmpfs /app-cache tmpfs rw,relatime,size=7168k 0 0
none /dev/cpuctl cgroup rw,relatime,cpu 0 0
/dev/block/mmcblk0p9 /system ext4 rw,noatime,nodiratime,barrier=1,data=ordered
/dev/block/mmcblk0p7 /cache ext4 rw,nosuid,nodev,noatime,nodiratime,barrier=1,
/dev/block/mmcblk0p1 /efs ext4 rw,nosuid,nodev,noatime,nodiratime,barrier=1,da
nil /sys/kernel/debug debugfs rw,relatime 0 0
/dev/block/mmcblk0p10 /data ext4 rw,nosuid,nodev,noatime,nodiratime,barrier=1,
/dev/block/mmcblk0p4 /mnt/.lfs j4fs rw,relatime 0 0
/dev/block/vold/179:11 /mnt/sdcard vfat rw,dirsync,nosuid,nodev,noexec,noatime
# dd if=/dev/block/mmcblk0p10 of=/sdcard/DataPartition.dd
dd if=/dev/block/mmcblk0p10 of=/sdcard/DataPartition.dd
4194304+0 records in
4194304+0 records out
2147483648 bytes transferred in 447.716 secs (4796530 bytes/sec)
```

Fig. 1 Kernel information of rooted android phone

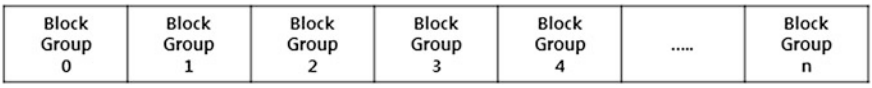

Fig. 2 Layout of ext4 file system

GDT(Group Descriptor Table), block bitmap, inode bitmap, inode table, journal log area, and data blocks. The layout of ext4 file system, block group 0, 1 and 2 are exemplified in Fig. 3; from block group 3, block group 1 and 2 are shown up in turns [1].

Super block is similar to boot record of NTFS file system and it stores overall information of file system, e.g., block size, number of blocks, number of block group, number of inode per block group, size of inode structure and so on. This super block area is located in the first block of block group and its size is 1 KByte [1].

GDT—Group Descriptor Table is made up of several Descriptors(32 Bytes). Descriptor exists as same as the number of block groups and each descriptor has offset of block bitmap, inode bitmap, and inode table in each block group. Therefore, when analyzing super block and GDT, we can obtain the whole information of file system [1].

Block bitmap is represented by one bit and check whether block is allocated or not. Namely, one block is mapped by one bit. Inode bitmap follows the same process as block bitmap. In case of allocated block or inode, bit is set as "1". Thus, file system manages allocation status of both block and inode by using bitmap [1, 2].

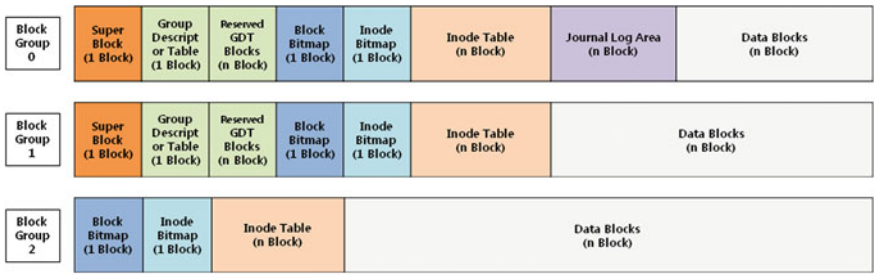

Fig. 3 Layout of block group

Inode table is allocated to each data(directory or file) in file system and meta data is recorded in this table, e.g., accessed time (Atime), inode change time (Ctime), modified time (Mtime), delete time (Dtime), create time (Crtime), file mode, file size, extents structure and so on. Super block shows the information of size of inode table [1, 2].

Directory entry is stored in data blocks and it record information of data stored in directory as an entry form, e.g., inode number, length of entry, length of file name, file type and file name [1, 2].

Only the first block group has journal log area and journal logs about transaction exist in this block. There is journal super block in the first block of journal log area and it records overall information of journal log. Many of journal logs are shown up from the next block, and there are descriptor block, metadata blocks, commit block, and revoke block included.

3 Forensic Analysis of Journal Log Area

3.1 Components of Journal Log Area

Every data in journal log area is recorded in the way of big endian. In addition, the components of journal log (e.g., journal super block, descriptor block, commit block, revoke block) have a journal header (12 Bytes). See Table 1. For structure of journal header below.

Entire information about journal log area such as size of journal block, number of blocks in journal log area, block number of the first transaction and so on are contained in journal super block.

Journal header in descriptor block records a sequence number about transaction. In the back of header, at least one more entry composed of block number and descriptor entry flags exists. There are not only block numbers of file system's metadata block but also block numbers of file system's journal log block (e.g., super block, GDT, block bitmap, inode bitmap, inode table, directory entry, commit block and revoke block) which will be changed by transaction. The value

Table 1 Structure of journal header [1]

Byte range	Description	Value	
0–3	Magic number	0 × C03B3998	
4–7	Block type	Description	Value
		Descriptor block	0 × 01
		Commit block	0 × 02
		Journal super block ver 1	0 × 03
		Journal super block ver 2	0 × 04
		Revoke block	0 × 05
8–11	Sequence number	Variable	

Table 2 Descriptor entry flags [1]

Flags	Description
0 × 00	The first entry
0 × 01	Journal block is wrong
0 × 02	UUID is as same as previous entry
0 × 04	Block is erased by transaction
0 × 08	The last entry (if there is only one entry existed)
0 × 0A	The last entry (if more than one entry existed)

of descriptor entry flags is displayed in Table 2. UUID comes after the first entry [4].

In metadata blocks, super block, GDT, block bitmap, inode bitmap, inode table, directory entry, descriptor block, commit block and revoke block are stored as a unit of block (such as snapshot). These blocks are original data blocks that are matched with block numbers in descriptor block. Therefore, by comparing blocks in metadata blocks and blocks matched with block numbers in descriptor block, we could know what changes have happened by transaction.

Commit block is placed in the last block of journal log. This commit block can be existed in metadata blocks; however, in case that sequence number of header in this block is equal to the sequence number of header in descriptor block, commit block becomes journal log itself. In $0 \times 34 - 0 \times 37$ Bytes, the timestamp information about commit performed is recorded as Unixtime (UTC 0) format so that we could make up timeline that occurs transaction with this information.

Revoke block is also located in the last block of journal log as same as commit block. Sequence number and block number related to revoke are stored in this block.

3.2 Data Recovery Using Journal Log Area Data

In order to carry out data recovery by using journal log, the information of journal super block is requirement. Through the information of size of journal block and number of journal log area block, we need to figure out the size of journal log area.

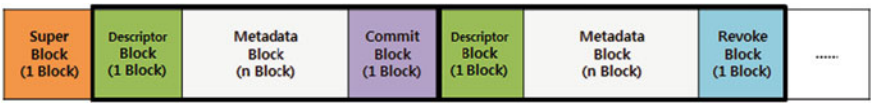

Fig. 4 Layout of journal log area

In the second place, we need to extract every journal log from journal log area while analyzing block as a unit with this information. Here are the steps of data recovery through journal log area.

3.2.1 Extraction of Journal Log

(1) Move from journal super block until finding descriptor block. Moving distance is as same as the size of journal block.
(2) One journal log start from the block we found descriptor block.
(3) Obtain sequence number from journal header of descriptor block and block numbers from entry of descriptor block. The sequence number of descriptor block is the sequence number of transaction regarding journal log.
(4) Move to data blocks pointed by descriptor entries and extract all blocks. The range of extraction is as same as block size of file system.
(5) After completing extraction, move to the next journal block to find metadata block.
(6) Extract all blocks from metadata blocks until we find commit block or revoke block which has sequence number like transaction. When finishing metadata block extraction, find commit block or revoke block which matches on journal log. Then extract it and complete extraction of one journal log. Overall layout of journal log is shown in Fig. 4.

3.2.2 Analysis of Journal Log and Data Recovery

(1) If we extract block of block bitmap, inode bitmap, inode table, directory entry at the same time in the 4, 6 process from extraction of Journal log, there is the possibility that transaction is related to deleted or modified file
(2) Compare hex values of blocks extracted from both descriptor and metadata blocks at one to one. See Fig. 5.
(3) If length of entry is changed in particular file after comparison of two directory entries, it means that transaction is took place.
(4) Check the deleted block number throughout the comparison of two block bitmaps, and extract the deleted block. This process, in other words, is called as data recovery.

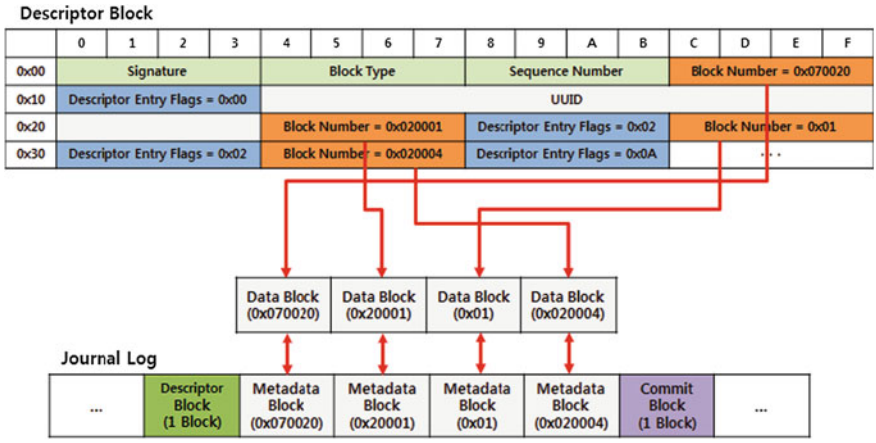

Fig. 5 Comparison blocks matched with block numbers in descriptor and metadata blocks

3.3 Experiment of Data Recovery Using Journal Log

In an experiment on data recovery using journal log, below equipment and tool were utilized: Samsung Galaxy S II, Google Nexus S android smartphone, WinHex by X-Way Software Technology AG [5], HexCmp2 by Fairdell Software [6], Encase7 by Guidance Software [7], Sqlite Expert Personal Edition by Bogdan Ureche [8]. Recovery target by using journal log data is a "db.db" file which store user data in "dropbox" application.

In the first place, we deleted "dropbox" application from android phone, then obtained DD image from data partition using ADB procotol. Afterwards, we attempted to recovery "db.db" file in the way of data recovery process which suggested in the second paragraph of Chap. 3.

Proceeding from what has been looked above, we could find the journal log related to "db.db" by extracting journal log from journal log area. By using HexCmp2 [6], we could compare hex values of two directory entries extracted from descriptor block and metadata blocks. As seeing the following of the values of changed entry length in Fig. 6, we found that "db.db" file is changed.

Figure 7 is the result of comparison of hex values in two extracted block bitmaps. We can see 2 bytes are changed. More detailed information of change is found in Fig. 8. We could realize total 10 bits are changed. Four bits out of 10 bits are changed 1 into 0 while 0 becomes 1 in the rest 6 bits. It means that 4 allocated block becomes unallocated block and other 6 blocks out of 10 becomes allocated.

Figure 9 is the result of 4 unallocated blocks that is confirmed with the function of disk view in EnCase7 [7].

Our results have confirmed that the internal data in this unallocated block is a "db.db" file; moreover, we recovered a "db.db" file which deleted "dropbox" application's user data are included. This way of recovery process enables us to

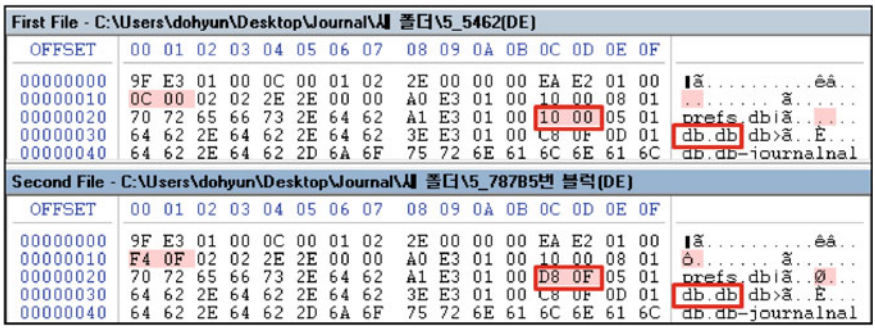

Fig. 6 The result of the comparison of two directory entries by using HemxCmp2 [6]

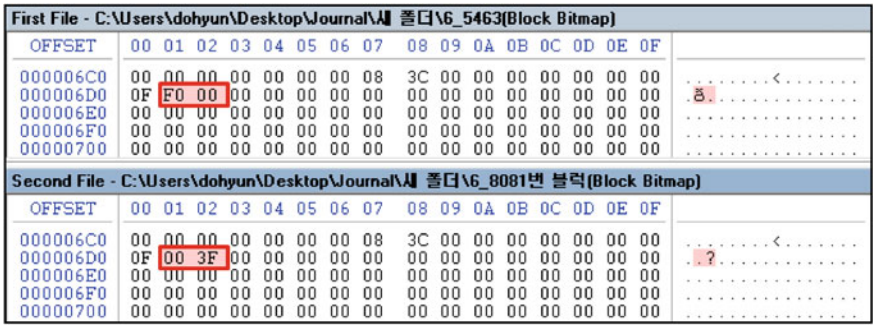

Fig. 7 The result of comparision of two blocks bitmap by using HexCmp2 [6]

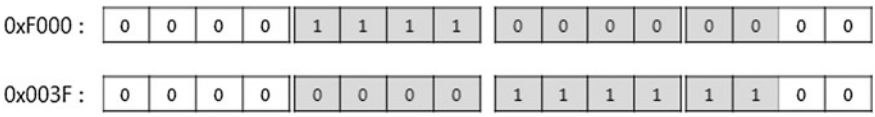

Fig. 8 In this conversion, 2 bytes hex values is 16 bits

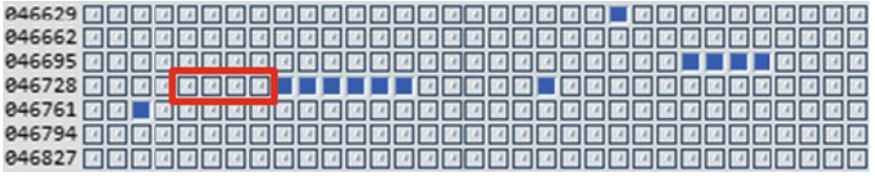

Fig. 9 The result of confirming the physical location of four changed blocks by using EnCase7′s diskview function [7]

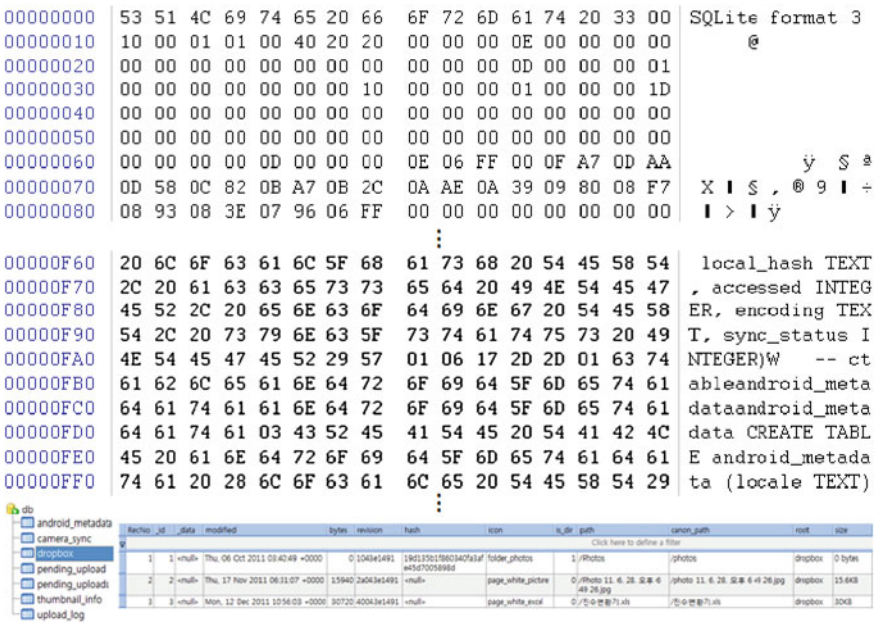

Fig. 10 The result of confirming the recovered file("db.db") by using WinHex [5] and SQLiteExpert [8]

extract appointed block which recovery is needed rather than carving that requires us to investigate all of unallocated area. Comparing carving which provides us just file data from extraction, we could also obtain information of file name and timestamp about recovery target. Therefore, it is very useful and time-saving that we could use in digital forensic investigation from various quarters (Fig. 10).

3.4 Analysis of User Actions Using Journal Log Area Data

As looking at the previous extraction process of journal log, we could obtain the information of data that is changed by transaction. First of all, through directory entry, we can take not only type of data(directory or file) out of journal log area data, name of data, timestamp through inode table, but also the location of deleted data block from block bitmap. Furthermore, commit block enables us to take the time when sequence number of transaction and commit occurred.

In sum, by making up timeline as well as integrating information from above process, we could analyze what and when a user has done with android phone. For this reason, Analysis of user actions on android phone analyzing ext4 in journal log is very important in terms of digital forensic investigation.

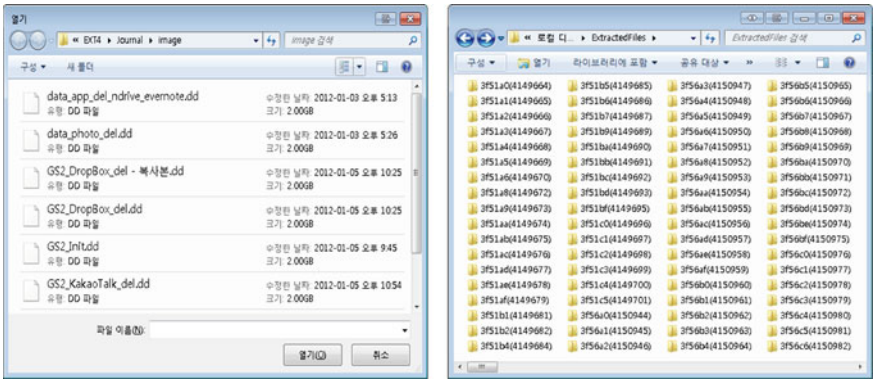

Fig. 11 UI of JDForensic inputted from imaging file and result of operating

4 Tool for Journal Data Analyzer

This paper has attempted to find the ways of data recovery and analysis through journal log. Even if it produces successful result from experiments, this process is time and effort consuming work. Thus, we develop the tool which called JDForensic, Journal Data Forensic, so as to provide data recovery and analysis of user actions.

JDForensic runs as journal data analyzer for digital forensic investigation and is performed in the process as follows; inputting image file; extracting and analyzing journal logs; creating new folder per sequence number of transaction; outputting extracted file and time from each folder. The following Fig. 11 is UI inputting image file and extracted folders.

5 Conclusion and Future Work

As ext4 file system is widely used in android phone and Linux system, more forensic research is needed on ext4 in digital forensic perspective.

As we have seen, the main purpose of this paper is to analyze journal log area in ext4 file system and to come up with faster and more accurate way of data recovery rather than carving. Hence, rather than utilizing carving in the process of data analysis, the way of data recovery through journal log area has advantage of analyzing data in a short space of time at the first stage of digital forensic investigation. Through time information of journal log, we also finding out when and what a user has done with android phone.

Unfortunately, there needs to be a continuing efforts to sort out transaction from user actions and operating system since both of transactions are mixed up in journal log area. Journal log area also has restriction on the amount of journal log;

however, we found that the most recent user actions (for approximately 1 day) can be analyzed throughout the result of experiments.

Up to know we have looked at the journal log area and developed JDForensic as a forensic tool by applying the way we proposed in this paper. JDForensic tool will be usefully utilized in digital forensic investigation of android phone; further studies on JDForensic tool will bring us more effective and accurate analysis of journal log data.

Acknowledgments "This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency)" (NIPA-2012-C1090-1101-0004)

References

1. Brian Carrier: "File System Forensic Analysis", Addison Wesley Professional (2005)
2. SANS Computer forensics and Incident Response Blog: "Understanding EXT4 (Part 1–4): Extents, Timestamps, Extent Trees, Demolition Derby", http://computer-forensic.sans.org
3. Blog about android phone rooting: Blocg of Tegrak, http://psp master.tistory.com
4. Narváez, G.: "Taking advantage of Ext3 journaling file system in a forensic investigation", SANS Institute Reading Room (2007)
5. X-way Software Technology AG homepage, http://www.x-ways.net
6. Fairdell Software homepage, http://www.fairdell.com
7. Guidance Software homepage, http://www.guidancesoftware.com
8. SQliteExpert homepage, http://www.sqliteexpert.com