TELKOMNIKA, Vol.16, No.2, April 2018, pp. 776~794
ISSN: 1693-6930, accredited **A** by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v16i2.8230

File Reconstruction in Digital Forensic

Opim Salim Sitompul*, Andrew Handoko, Romi Fadillah Rahmat
Department of Information Technology, Universitas Sumatera Utara,
Jl. Universitas No 9A, Kampus USU, Medan, Indonesia, telp/fax. 62618228048
*Corresponding author, e-mail: opim@usu.ac.id, andrewhandoko@rocketmail.com,
romi.fadillah@usu.ac.id

Abstract

File recovery is one of the stages in computer forensic investigative process to identify an acquired file to be used as digital evident. The recovery is performed on files that have been deleted from a file system. However, in order to recover a deleted file, some considerations should be taken. A deleted file is potentially modified from its original condition because another file might either partly or entirely overriding the file content. A typical approach in recovering deleted file is to apply Boyer-Moore algorithm that has rather high time complexity in terms of string searching. Therefore, a better string matching approach for recovering deleted file is required. We propose Aho-Corasick parsing technique to read file attributes from the master file table (MFT) in order to examine the file condition. If the file was deleted, then the parser search the file content in order to reconstruct the file. Experiments were conducted using several file modifications, such as 0% (unmodified), 18.98%, 32.21% and 59.77%. From the experimental results we found that the file reconstruction process on the file system was performed successfully. The average successful rate for the file recovery from four experiments on each modification was 87.50% and for the string matching process average time on searching file names was 0.32 second.

Keywords*: digital forensic, file undelete, file recovery, Aho-Corasick algorithm, finite state automata*

1. Introduction

A file could be used as an authentic evident in certain criminal cases. Digital evidents are data stored or transmitted using a computer to support or to deny a criminal act. In this case, a digital file shows some important elements of a criminal act that could be used either as a motive or as an alibi [1]. Accordingly, a criminal will try to eliminate files that can be used as evidence of his criminal acts simply by deleting the file from the storage media. File deletion is in fact only a deletion of the file reference from system Table [2] such that clusters where the file contents are allocated become unallocated spaces. Unfortunately, data in an unallocated space could potentially be lost if the same location is later overridden by another data. In this case, the act of deleting files from storage media will bring some difficulties and the file recovery becomes more difficult. Harder file recovery attemps will in turn hinder the digital forensic investigator in gathering digital evidences.

Luckily, digital evidence files that had been removed from a file system are still be able to be recovered. As mentioned earlier, the act of deleting a file from the file system is in fact only changing the reference of the file on the Master File Table (MFT) that results in the clusters occupied by the file are marked as unallocated spaces. Therefore, it is still possible to reconstruct the deleted file since the file contents are still available on the storage medium as long as there are no overwrite processes on the file, no thorough deletion, or hard disk wiping on the media are performed [1]. Although a deleted file could no longer be accessed by the file manager, using a file undelete approach the deleted file could be restored. One algorithm to perform a file recovery is Boyer-Moore algorithm [3], which has time complexity of O(*mn*) in searching phase.

Even though the string-matching Boyer-Moore algorithm has a linear time complexity, the speed could still be improved using Aho-Corasick algorithm which is a string searching algorithm with linear time complexity of O (*n*+*m*+*z*) with *n* serves as the number of patterns, *m* as the length of the text used in the search, and *z* is the number of corresponding outputs or number of pattern occurrences [4]. This algorithm is a dictionary adjustment algorithm that places elements in a finite string set and adjusts all the patterns simultaneously. Aho-Corasik

Received October 5, 2017; Revised January 28, 2018; Accepted February 12, 2018

algorithm will first create a tree-like automata engine, called trie. Trie is an ordered tree data structure that is used to store dynamic sets or associative arrays where the existing key is usually a string. A trie has many advantages over the binary tree [5] and can also be implemented to replace hash tables. A trie has an additional link between the internal nodes of the keyword or the existing patterns. This additional link enables rapid transitions when there is a failure in the pattern matching process, by which the automata can move to another trie branch that has similar prefix without the need for backtracking. Aho-Corasick algorithm has been applied to solve numerous problem such as signature-based anti-virus application [2], set matching in Bioinformatics [4], structural-to-syntactic matching for identical documents [6], searching of text strings on digital forensics [7] and text mining [8].

In terms of recovering files, research work by [3] implemented carving method using Boyer-Moore algorithm to recover deleted files. In accordance with the results, some issues such as lengthy processing time and high-capacity storage were faced in the carving process. Over 1.1 million files with total size of 250GB were produced in the carving process of 8GB target disk, despite a very large amount of false positive. In conclusion of the research, Boyer-Moore algorithm was not recommended to be implemented in matching process of file header and footer, which is O(mn). In 2010, [9] conducted research to reconstruct MP3 file fragment using *Variable Bit Rate* (VBR). The proposed method was successfully increased the success rate in finding the correct file fragment to be reconstructed. The increment percentage for high quality MP3 file was 49.20–69.42%, 1.80–3.75% for medium quality file, and 41.2–100.00 % for low quality file. The increasing rate in finding fragment from file will improve the performance of carving process. Another study by [10] conducted in 2011 applied a carving method for multimedia file. The result showed that the method was able to successfully recover multimedia files of MP3, AVI, and WAV for continuously allocated files. Although the file was allocated at times, it could still be identified through its characteristics after the recovery process. Despite the difficulty in recovering a compressed multimedia file saved in NTFS, it could still be restored using the carving method.

Recent works on media file forensic, such as audio, photo, and video were also found in [11],[12]-[13], respectively, as well as digital forensic on Hadoop [14]. In [11], an audio forensic on identical microphones was conducted using statistical based method, while in [12] an algorithm on photo forensic was proposed in order to detect image manipulation using error level analysis. Forgery detection on video inter-frame had also been conducted in [13] for survellance and mobile recorded videos. As popularity of big data analysis has been flourish in recent days, [14] conducted a study on digital forensic in Hadoop.

This paper is an extended version of our previous publication [15], while the initial stage of research had also been described in [16]. Our previous research works related to this study was described in [17] which encompassed the file type identification using Distributed Adaptive Neural Network that was introduced and derived from [18-20].

The objective of this research is to restore all the deleted files from file system using file undelete approach and Aho-Corasick algorithm so that the file could be analyzed to check whether it is undamaged file or file that containing fragments of other files. The scope of this research is focused on hard disk with NTFS file system that was checked and utilized in the recovery process. Furthermore, it is required that the storage media does not experience any process of wiping or data overwriting, that will damage the Master File Table.

2. Research Method

The proposed method for this research could be described in four stages, such as disk imaging, accessing MFT, file type identification and corruption check, and file reconstruction consisting of undelete, verification and analysis steps. Figure 1 depicts the general architecture of every stage performed to reconstruct deleted files in a file system. Each stage could be describe in detailed steps flow as follows.

Step 1. Duplication of storage media contents (disk imaging) to obtain duplicates of storage media that are identical to the actual storage media;
Step 2. Accessing and reading MFT records to search records from all existing files and directories;
Step 3. Metadata extraction from MFT record using parsing on MFT record;

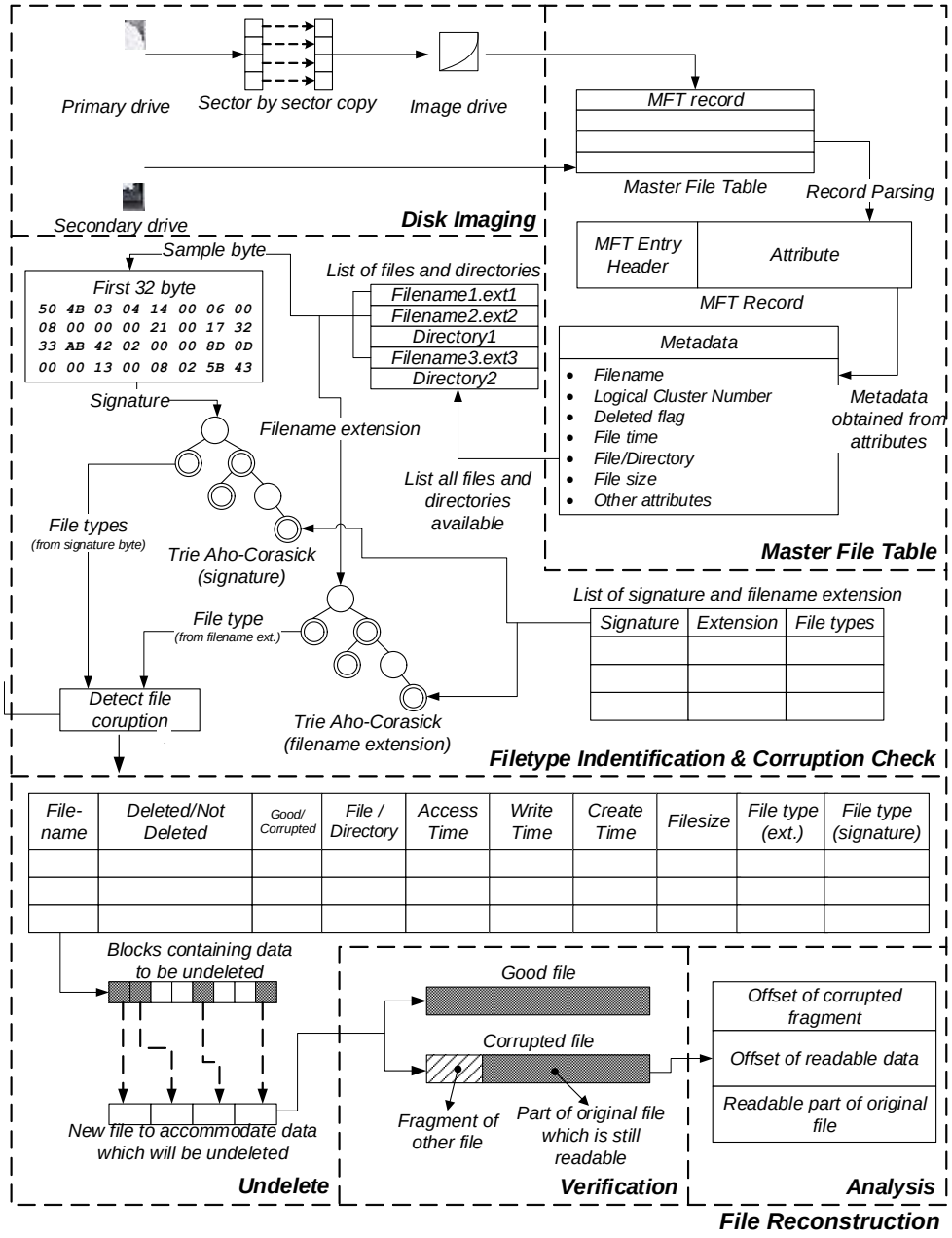

Figure 1. Proposed method-general architecture

Step 4. Performing parsing on file name to obtain filename extension;
Step 5. Taking the first 32 bytes of the cluster occupied file as sample;
Step 6. Building trie using Aho-Corasick algorithm based on actual signature and file extensions;
Step 7. Identifying file type based on signature and filename extension;
Step 8. Comparing the results of file type identification based on filename extension with the result of signature based identification to see whether the file was damaged;
Step 9. File registration along with its details, such as the file type (based on signature and filename extension), timestamp, file condition, and other types of information;
Step 10. File reconstruction based on metadata obtained from MFT, verification of the reconstructed file by opening the file then check for the signature so that it can be identified whether or not the recovered file experienced any damage; analyze the damaged files and read the readable information of the file;

Step 11. File recovery on the file system deleted files could then be performed after all the previous steps were done.

After performing the steps in the proposed method, the developed program will be able to recover deleted files on file system and to select any file that need to be reconstruct based on the given keywords. Each step that was performed will be described in detail in the following sub-sections.

2.1. Disk Imaging

This stage will duplicate the content of the storage media in sector level so that the duplicate acquired is identical to the original storage media including the boot sector and the MFT. From this process the acquired duplicate will be used to access and to read the MFT record. This stage could be optional if the storage media used is a secondary drive as it will not be accessed directly by the operating system or other applications. Figure 2 shows the disk imaging scheme.

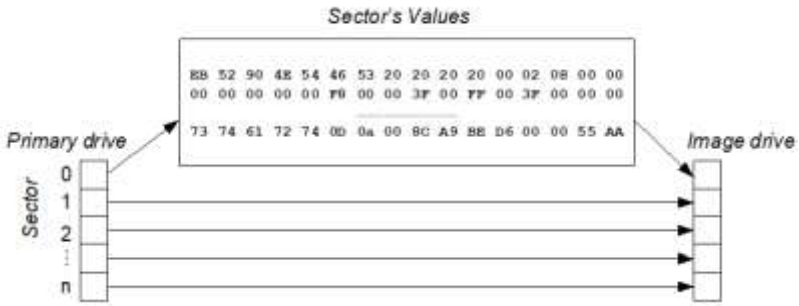

Figure 2. Disk Imaging Scheme

This research used a 4GB secondary storage media with NTFS file system (the effective size is 90% from the storage media size that is 3.60GB or 3,873,783,808 bytes) with cluster size of 4KB (4096 bytes). There were 56 files from various file types with a total size of 3.54GB (3,797,409,792 bytes). The calculation of CRC-32 value was performed on each file to be used as comparison variable in verification process.

2.2. Master File Table (MFT)

In this stage, the cluster number containing the MFT was read from the 0^{th} boot sector, which was located on 0x30 offset and 8 bytes on length using little endian system. Each record in the MFT was accessed to read information of each file and directory in the storage media. Each record went through the parsing process to break up each record based on MFT entry header. Attribute header contains information about type, size, and name of the file and also attribute value pointing to actual data.

Each file and directory contained in the storage media has some information stored in the MFT record. The MFT record provides information such as:

1. The type and condition of the records. This information is attained from the offset 0x16 values of 2 bytes in MFT entries.
   a. If the value is 0x00 then the record is a record for file and it can no longer be used (the files has been deleted from file system)
   b. If the value is 0x01 then the record is a record for file and it is still used (the file is still listed in the file system).
   c. If the value is 0x02 then the record is a record for directory and it can no longer be used (the directory has been deleted)
   d. If the value is 0x03 then the record is for directory and it is still used (the directory is still listed in the file system).

2. File condition. The file condition is identified by comparing the result of file identification based on filename extension and the signature. The deleted files could be in several conditions, i.e.:
   a. Good. The file is considered in a good condition if after deleted, the cluster occupied by the file is not used by another file.
   b. Damaged. If after deleted, the original cluster is occupied by another file. Then the file is damaged.
   i. If the override file is larger than or equal to the original file, then the original file will be completely overwritten. The completely overwritten files will have content differences from the original file.
   ii. If the override file is smaller than the original file, the original file will be partially overwritten. If a file is partially overwritten, then some of the information from the original file is still readable. Fragment of the original file may contain information that can still be recovered.
3. File size (in bytes).

Parsing process is mandatory to obtain file metadata for the reconstruction process such as filename, number of logical cluster occupied by the file, flag to determine the deleted file, and other information. Based on the metadata obtained, all files with deleted flag will be listed. An MFT Record consisting of hexadecimal numbers is shown in Table 1.

Table 1. MFT Record

Offset	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	46	49	4C	45	30	00	03	00	57	34	80	00	00	00	00	00
10	03	00	01	00	38	00	01	00	E0	01	00	00	00	04	00	00
20	00	00	00	00	00	00	00	00	06	00	00	00	03	00	00	00
30	03	00	00	00	00	00	00	00	10	00	00	00	48	00	00	00
40	00	00	18	00	00	00	00	00	30	00	00	00	18	00	00	00
50	24	CC	73	D0	62	D9	CF	01	24	CC	73	D0	62	D9	CF	01
60	24	CC	73	D0	62	D9	CF	01	24	CC	73	D0	62	D9	CF	01
70	06	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
80	30	00	00	00	68	00	00	00	00	00	18	00	00	00	01	00
90	50	00	00	00	18	00	01	00	05	00	00	00	00	00	05	00

The MFT entry parsing process is as follows:

a. Offset 0x00 with length of 4 bytes is magic number “FILE”.
b. Offset 0x06 with length of 2 bytes is number of fixup array, which is 0x00 03=3 arrays.
c. Offset for the first attribute obtained from offset 0x14 with length of 2 bytes with little endian reading of 0x00 38
d. Offset 0x16 with length of 2 bytes is flag, since the value is 0x00 01, so this record is the record for files. Then parsing is performed on the attributes located in MFT record.
e. The first attribute found on the offset 0x00 38 of the record with the first 4 bytes is a marker of the attribute type of 0x00 00 00 10. This attribute is an attribute containing $STANDARD_INFORMATION or standard information.
f. The next 4 bytes are the length of the attribute 0x00 00 00 48=72 bytes.
g. The next 1 byte is a non-resident marker flag, since the value is 0x00 then the attribute is a resident attribute.
h. The attribute has a content size of 0x00 00 00 30=48 bytes and starts at offset 0x00 18=24 (offset of attribute).

Every record will go through parsing process in order to obtain metadata to perform reconstruction process, such as file name, logical cluster number occupied by the file, flag indicating deleted files, flag indicating a file or directory record, timestamp, and other information.

2.3. File Type Identification and Corruption Checking

In this stage, the files contained in the storage media were identified to determine whether or not the files were corrupted. File type identification was performed first based on filename extension and file signature. Filename extension was derived from the file name while

file signature was obtained from the file header by taking sample of the first 32 bytes as shown in Figure 3.

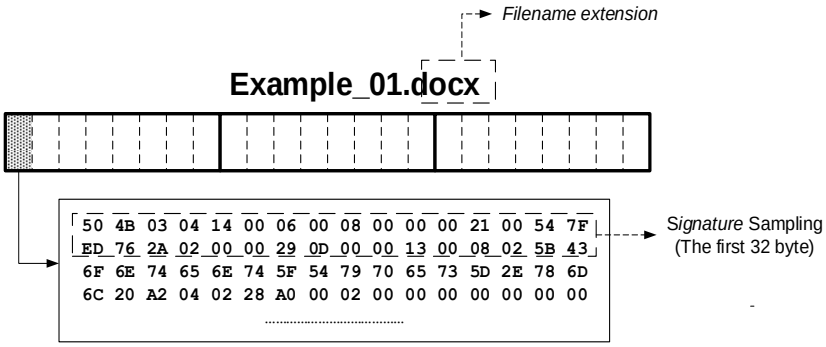

Figure 3. Filename extension and signature sample

Filename extension and signature of the actual file are information required to identify the file type. The filename extension and signature to be utilized are shown in Table 2.

Table 2. List of Filename Extension and Signature to be utilized

No	*Filename Extension*	*Signature*	Description
1.	DOC, DOCX, PPT, PPTX, MSI, VSD, XLS, XLSX	• 50 4B 03 04 14 00 06 00 • D0 CF 11 E0 A1 B1 1A E1	Document or Microsoft file
2.	PDF, FDF	• 25 50 44 46	Adobe Portable Document Format
3.	JPEG, JPG	• FF D8 FF E0 00 10 4A 46 49 46 00 01 01 • FF D8 FF E0 • FF D8 FF E1 • FF D8 FF E8 • FF D8 FF	JPEG file
4.	PNG	• 89 50 4E 47 0D 0A 1A 0A	PNG file
5.	GIF	• 47 49 46 38 39 61 4E 01 53 00 C4	GIF file
6.	MP3	• 49 44 33	MP3 file
7.	MKV	• 1A 45 DF A3 93 42 82 88 6D 61 74 72 6F 73 6B 61	Matroska file
8.	MP4, M4V	• 66 74 79 70 33 67 70 35 • 66 74 79 70 4D 53 4E 56 • 66 74 79 70 6D 70 34 32	MPEG-4 video file
9.	EXE	• 4D 5A	*Executable file*
10	RAR	• 52 61 72 21 1A 07 00 • 52 61 72 21 1A 07 01 00	Compressed archive
11	ZIP, JAR	• 57 69 6E 5A 69 70 • 50 4B 03 04 • 50 4B 05 06 • 50 4B 07 08	*Zip archive*

The information in Table 2 would be converted into two types of tries, namely filename extension trie and signature trie. An example of those two tries are shown in Figure 4.

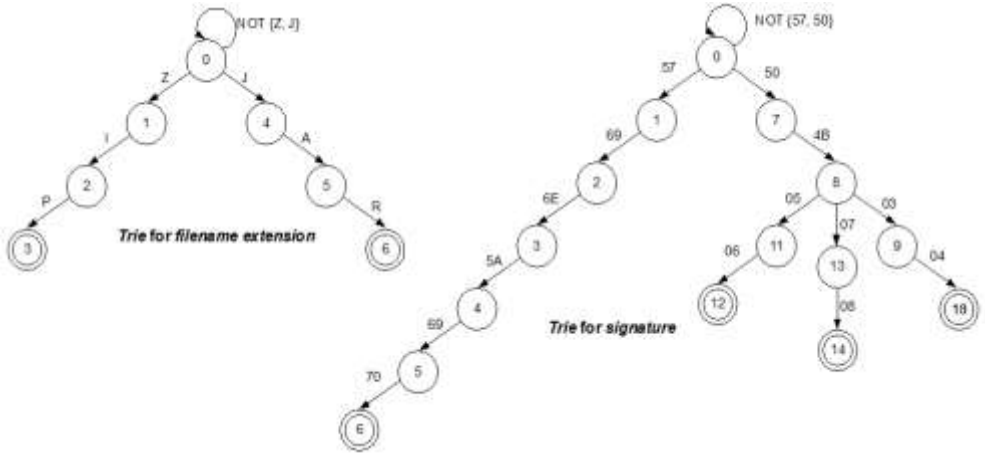

Figure 4. Tries for filename extension and signature

After all filename extensions and signatures information are converted into tries then the file type identification was performed. Aho-Corasick algorithm identifies the filename extension and signature based on trie for filename extension and trie for signature, respectively. The identification process generates two identification results, and the two results are compared to determine the file condition. The comparison of the two results and the output of the file condition are shown in Table 3.

Table 3. The Comparison of Filename Extension and Signature Identification Results and Files Condition

No	*File type* (*filename extension*)	*File type* (*signature*)	Comparison of the two identification results	Condition
1.	Identified	Identified	Match	Good
2.	Identified	Identified	Not match	Damaged
3.	Unidentified	Identified	Not match	Damaged
4.	Identified	Unidentified	Not match	Damaged
5.	Unidentified	Unidentified	-	Unknown

From Table 3 there are three types of file conditions, namely “Good”, “Damaged”, and “Unknown” for five comparison conditions, which could be described as follows.

a. If the file type is identified based on filename extension and signature and both identification results give the same result, then the file is still in “Good” condition.
b. If the file type is identified based on filename extension and signature but both identification results give different results, then the file is “Damaged”.
c. If the file type failed to be identified based on filename extension but successfully identified by signature, then the file is damaged. This condition can occur on files that have been experienced forgery or have been overwritten by other data.
d. If the file type is identified based on filename extension but failed to be identified by signature, then the file is damaged. This condition can occur on files that have been overwritten by other data.
e. If the file type failed to be identified based on both filename extension and signature, then the file condition is “Unknown”.

2.4. File Reconstruction

Metadata obtained from MFT records through parsing process is used to perform file reconstruction (recovery). Important information required for file reconstruction is the file name, Logical Cluster Number (LCN), and the size of the allocation. A complete file information collected from the file system is listed in Table 4.

Table 4. Information Used in Undelete Process

No	Information	Details
1.	*Filename*	The filename corresponds to the entry on MFT.
2.	*File/Directory*	Entry type, whether an entry for a file or directory. Obtained from flags on MFT records
3.	*Deleted*	Condition whether the entry has been deleted or is still in use. Retrieved from the flag in the MFT record.
4.	*Condition*	Condition of the file damage obtained from the identification and comparison of file types.
5.	*Write Time*	The time a file is written.
6.	*Create Time*	The time a file was created.
7.	*Access Time*	The time a file is accessed.
8.	*Signature Filetype*	File type based on the identification of first 32 bytes of file.
9.	*Filename Extension Filetype*	File type based on the identification of filename extension
10.	*Allocation Size*	The size of file allocation in storage media. This information will be used in file reconstruction.
11.	*Logical Cluster Number*	The number of clusters occupied by the file. This information will be used to read the data stored in the cluster in file reconstruction process.

Although to perform the file reconstruction only requires information in the form of filename, LCN, and file allocation size, but other information such as file condition (damaged or good), file type, timestamp, etc. should also be collected in order to help in selecting the files to recover.

2.4.1. Undelete Process

The undelete process is the first step of the file reconstruction as illustrated in Figure 5.

a. Creating a new empty file with the same name as the file name to be restored and the same size as the allocation size of the restored file.
b. Opening the LCN of the recovered file and reading the contents of the cluster stored in the buffer. The amount of data read from the cluster is the same as the amount of buffer size allocation. The contents of the buffer are then written into a new file in hexadecimal values. After the data are written into the empty file, the buffer are re-used to accommodate the next data. The data writing is proceed from the last offset of the written data. This process will continue iteratively until the contents of all clusters occupied by deleted files are moved to the new file.

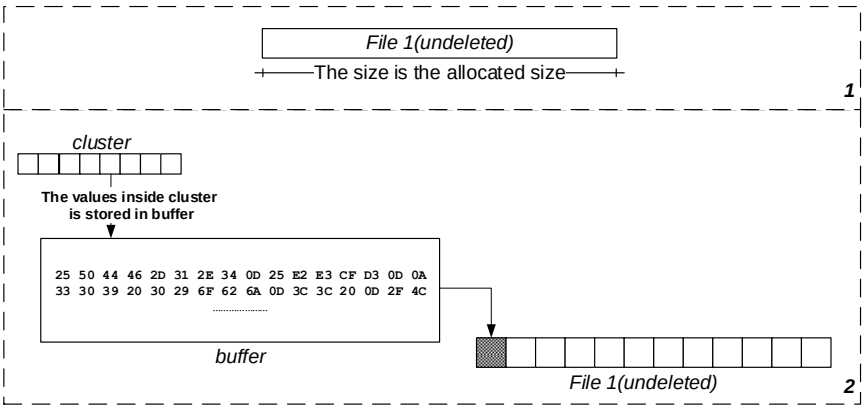

Figure 5. File reconstruction process at the time of recovery

2.4.2. Verification Process

Files recovered after deletion could be in several conditions, such as:

a. The file is restored properly. A file could be restored properly only if it is not overwritten by another file. Thus, if the file is restored, its condition is the same as the file condition before it was deleted

b. Files whose contents partially had been overwritten by other files. This condition can occur if after the file was deleted, the storage media was filled with another file smaller than the deleted file and the file uses the location of the deleted file clusters. If the deleted file was overwritten in the header section, then after the file has been restored, it can not be opened with the initial application. However, some of the file contents are still readable.
c. Files that are entirely overwritten by other files. This condition occurs if the cluster location originally occupied by the deleted file is filled by another file with the same size or larger than the file. If this file is restored, the contents will be different from the original file.

Based on the above conditions, the recovered files must go through a verification process. The verification process will determine whether or not the file is damaged. Verification is performed by opening the file with default application for that file. But if the file is damaged, then the hexadecimal value of the file will be read. The file will then be analyzed to obtain information from the file readable part.

2.4.3. Analysis Process

The analysis of the damaged file is conducted to determine whether the file is partially or completely corrupted. Entirely overwritten files will resulted in the recovered file is different from the actual file, while partially overwritten files are still have some information from the original file. Once it is known that the file is partially overwritten, procedure to read the remaining information from the original file can be applied. The steps are:

a. Determine the size of the data occupying the original file location. There are two ways could be taken i.e. by searching the footer of the override data or finding the hexadecimal value used by the operating system to fill the slack space (generally the value of hexadecimal 0x00 or NULL) as shown in Figure 6.

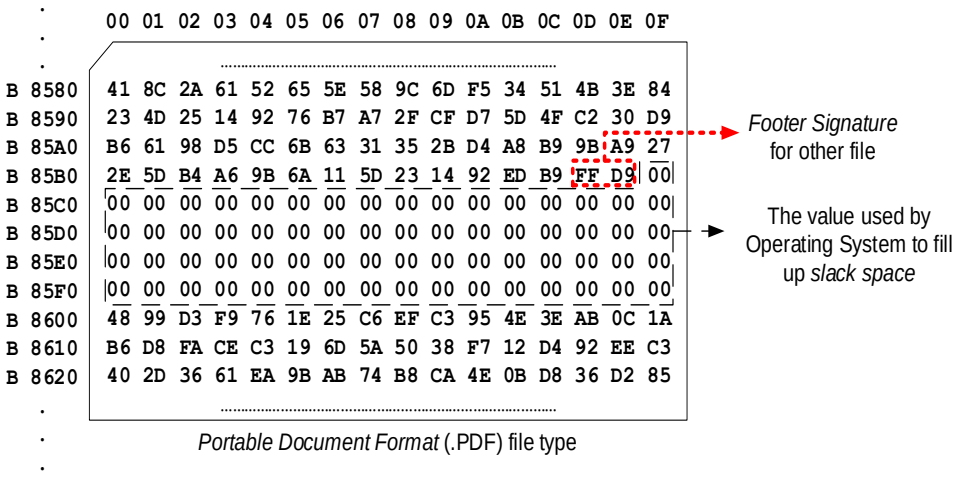

Figure 6. Analysis of damaged files

b. After the end of the override data is found, the size of the override data can be calculated by subtracting the final offset with the initial data offset.
c. Because files allocation in storage media are based on a clusters, the number of clusters used by the override data can be calculated by dividing the size of the override data by cluster size.
d. If n is the number of cluster used by the override data, then the readable part of the original file is in the $n + 1$ cluster or at offset $m + 1$ with m representing the last offset of the cluster n.

3. Results and Discussion

This section will describe the result obtained from the reconstruction process. There were 56 files used in this experiment with a total of 3.54GB (3,797,409,792 bytes). The recovery process were conducted on various file types including .docx, .pdf, .jpg, .png, and .exe files among others. From the 56 files, 55 file (98.21%) were successfully recovered with a total size of 3.52GB (3,781,166,380 bytes) or 99.71% from total size of deleted files.

In the testing process, the recovery of a file which is performed twice can result in different elapsed time. This is due to hardware condition such as processor speed, memory size, and access speed of the storage media. Even though the recovery is performed with the same hardware, differences in elapsed time can still happen as it might be caused by different processor load at the time the recovery process was performed. If the hardware factors (processor speed, memory size and speed, and storage media access speed) and data processing speed are considered constant for all files, then elapsed time is proportional to the size of the file to be recovered.

A line chart showing the relation between the elapsed time needed to recover files and the amount of bytes recovered is presented in Figure 7. The graph in Figure 7 shows that the elapsed time needed to recover file is proportional to the size of the file. As for the file types, it was discovered that they were not given significant impact on the elapsed time in the recovery process.

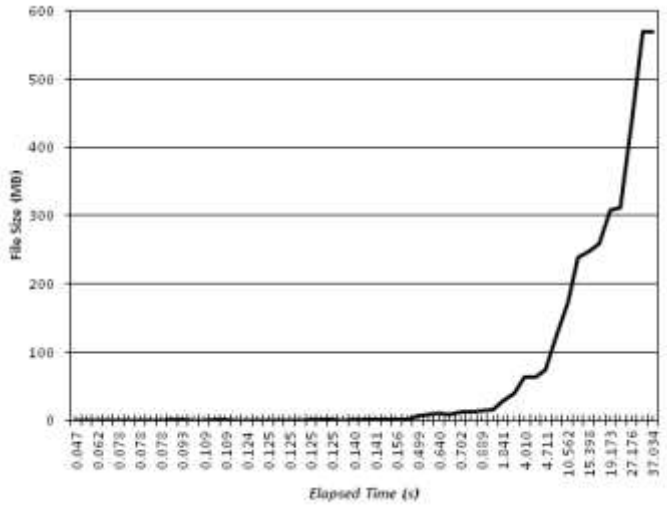

Figure 7. Relation between elapsed time versus the number of bytes recovered

Some factors that might affecting the speed of file recovery are hardware specification, such as processor speed, the size and speed of memory, and the speed of storage media access. Another factors that should be consider are processor load and the file size. Meanwhile, factor affecting the success rate of undelete process could be identifed such as condition of the MFT, size of the overwritten data on storage media, and size of the file to be recovered. A screenshot of an undelete process is shown in System Preview as in Figure 8.

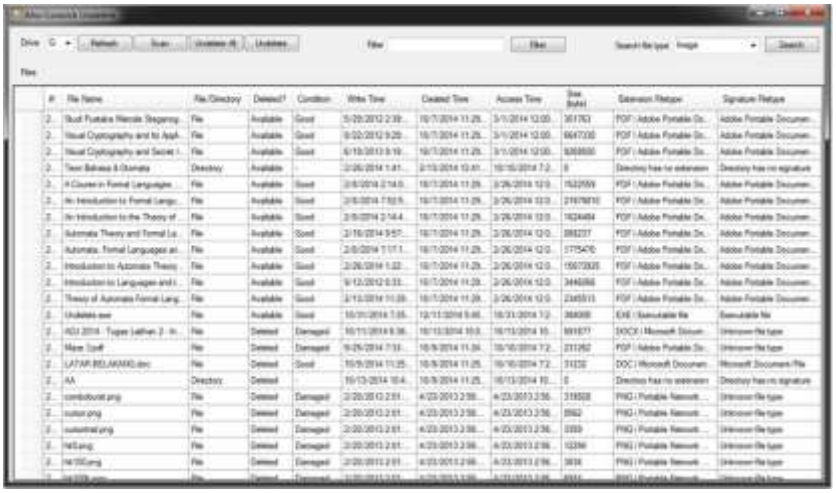

Figure 8. System preview of the undelete process

3.1. Result of Undelete Process

Analysis was performed based on the information obtained from damaged files in order to determine their sizes and readable parts. A DOCX file was used in this analysis process. Information obtained from damaged files is given in Table 5.

Table 5. Information of damaged files

Parameter	Value
File type based on filename extension	Microsoft Document (.DOCX)
File size	4.65 MB (4,884,115 byte)
File size with slack space (in storage media)	4.66 MB (4,886,528 byte)
(Sample of the file first 32byte)	F2 71 16 55 92 CE 0D C1 8F 13 45 0E 3F E6 D4 F9 01 0F E4 D3 2C F4 89 53 88 A0 58 44 C1 D5 F8 EB
	No known signature found
File type based on signature	Unknown
Examination result of the file damage	Damaged file
Sample of the file last 32byte	62 65 72 69 6E 67 2E 78 6D 6C **50 4B 05 06 00 00** **00 00 54 00 54 00 D3 16** **00 00 AA 6F 4A 00 00 00**
	Footer for Microsoft document file found

There was no known signature found at the beginning of the file, however footer for the DOCX file was found at the end of the file. The file was overwritten on its beginning by a fragment of another file whose signature value was not found in the signature trie, consequently the override file was unidentified.

Furthermore, the analysis process continues to find the value used by the operating system to fill the slack space in 0x00 sector and the footer of PDF file was found in the hexadecimal value of 0x00 from the offset of 0x167373 to 0x167FF0 and offset of 0x167E6D to 0x167E72 as illusrated in Figure 9. Thus, it was identified that the file was overwritten by the fragment from another file at the beginning. Once the condition of the file damage was identified, then the calculation of override data offset and the original file offset could be done.

The file had filename extension of DOCX and size of 4,884,115 bytes or from offset 0x00 to 0x4A8692. Then the value of the hexadecimal numbers contained in the file was traced. The value of hexadecimal numbers of 32 bytes (from offset 0x00 to 0x1F) found at the beginning of the file are "F2 71 16 55 92 CE 0D C1 8F 13 45 0E 3F E6 D4 F9 01 0F E4 D3 2C F4 89 53 88 A0 58 44 C1 D5 F8 EB ". From the hexadecimal values, no signature was found from the document file DOCX ("50 4B 03 04 14 00 06 00") and at the beginning of the file there

was no signature of a known file. Furthermore, offset 0x167E73 to 0x167FFF was filled with value "00", which was the value filled by the operating system to fill the slack space in the sector that has been populated with data. In addition, the hexadecimal values "0A 25 25 45 4F 46" were found at offsets 0x167E6D to 0x167E72 which were the signature for the footer of PDF file. Thus, the data from offsets 0x00 to 0x167E72 were data for PDF file fragment.

According to the results of searching process, information obtained are as follows:

a. Offset containing other file data=0x00 to 0x167E72 or equivalent to 1,474,163 byte.
b. Offset of slack space filled by operating system (hexadecimal value “00”)=0x167E73 to 0x167FFF.
c. Number of occupied clusters:
d. $\left\lceil \frac{\text{Data Size (byte)}}{\text{Cluster Size (byte)}} \right\rceil = \lceil 359.9 \rceil = 360$
e. The occupied size=number of clusters × cluster size=360 × 4,096=1,474,560 bytes (offsets 0x00 to 0x167FFF)
f. The data size of the uncorrupted old files are from offsets 0x168000 to 0x4A8692=3,409,555 bytes.

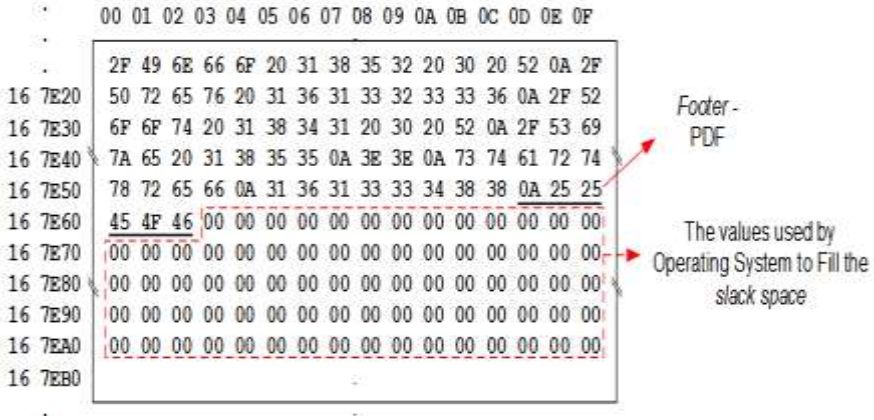

Figure 9. The value used to fill in slack space and footer from PDF FILE

Therefore, the number of clusters required to accommodate data of 1,474,163 bytes are 359.9 clusters. Because addressing is performed per cluster, the required number of clusters is rounded to 360 (equivalent to 1,474,560 bytes). Thus, the offsets used are 0x00 to 0x167FFF.

The next data of a readable EXE file is data starting from offset 0x168000 to 0x4A8692. The data size that overwrites the actual file data is 1,474,163 bytes (1,474,560 bytes when added with slack space) and the data size of the readable original file is 3,409,555 bytes. The analysis result of the data contained in a file is shown in Figure 10.

After the file went through the analysis process, the remaining information could be read from the uncorrupted original file. However, the readable information of the file depends on the level of file damage and the encoding method the file uses. Figure 11 shows that even though the file had been corrupted, some information from the file was still readable. The file in Figure 11 experienced damages in the header section. Although the header of the file has been overwritten by another file data, some information from the text file in ASCII encoding are still readable.

The readable information of damaged file is not only in the form of text. An image file can be stored in other files, such as Microsoft documents and Adobe Portable Document Format (PDF). Figure 12 shows how a JPEG file can still be read from the damaged documents (overwritten by HTML file at the beginning of the file)

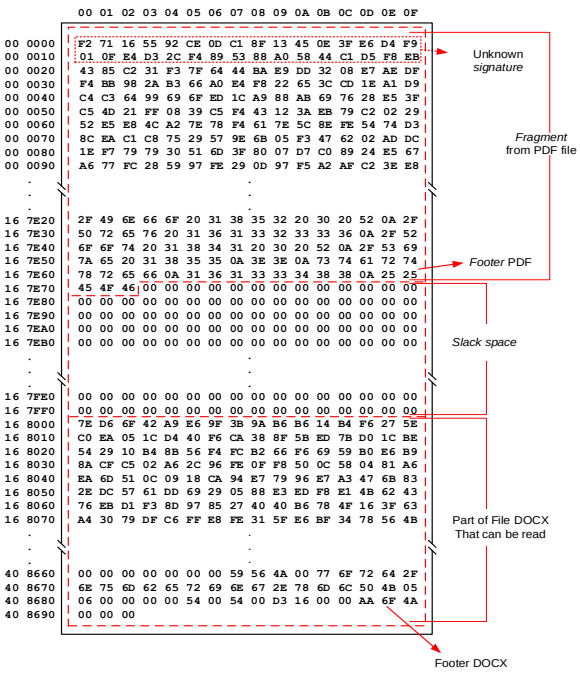

Figure 10. Analysis result of the data obtained in a file

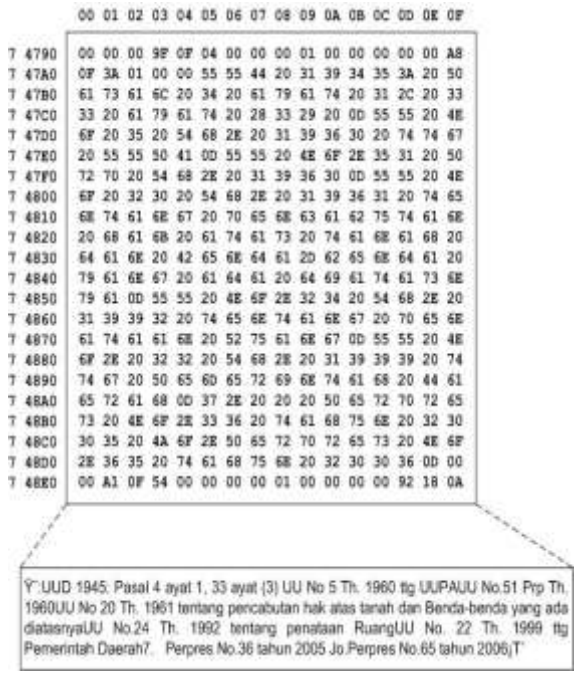

Figure 11. Readable information of damaged file

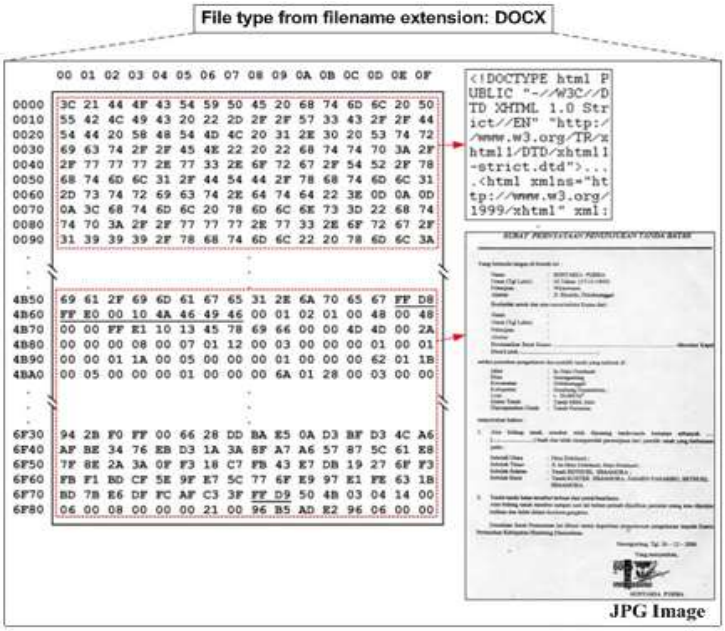

Figure 12. Image file recovery of damaged document file

To recover an image file from a damaged document file, the document file containing the signatures for the header and the footer of the image file were moved to an empty file and were written in the form of hexadecimal values. Thus, the file was identified as an image file and can be opened with an application for image files.

The readable information of the damaged document depends on the types and damage level of the documents, some information are still readable in spite of the file damage. In the

example, one of the documents with readable contents was a PDF document, even though it had suffered damage in which some data in the file was overwritten by another file as shown in Table 6.

Table 6. Information of damaged PDF document

Parameter	Value
File type based on *filename extension*	Adobe *Portable Document Format* (.PDF)
File type based on *signature*	Unknown.
Result of file damage test	File damaged.
File size	1,03 MB (1.081.946 bytes)
File size with slack space (in storage media)	1,03 MB (1.085.440 bytes)
Sample of the first 32 bytes	B9 BC BD D4 6E 2C ED 27 9B 31 25 AE 97 6B 3C C9 00 BE F3 E6 6B 78 51 63 49 1E 6B 74 79 DE 40 20
	No signature found for the PDF header
Signature for the found footer	a. FF D9 (*Footer* JPEG) on *offset* 0xB85BD. b. 45 4F 46 0D (*Footer* PDF) at the end of the *file* (*offset* 0x108256).
Initial readable data file size	c. Based on the size difference with the override data size: 317 KB d. The actual size: 290 KB
Number of readable pages	104

The damaged PDF document were overwritten by a JPG image (JPG footer with value of 0x00 minus value for slack space, was found on offset 0xB85BD). The number of cluster overwritten by the JPG image is [184.35]=185 clusters. Therefore, the size of the overwritten data is 757,760 bytes and the unaffected data size is 1,081,946–757,760=324,186 byte=317 KBs. The data of 317 KB is not entirely readable. Of the 317 KB fragment size, readable data is 290 KB or as many as 104 pages. Some pages from the PDF documents are still readable as shown in Figure 13.

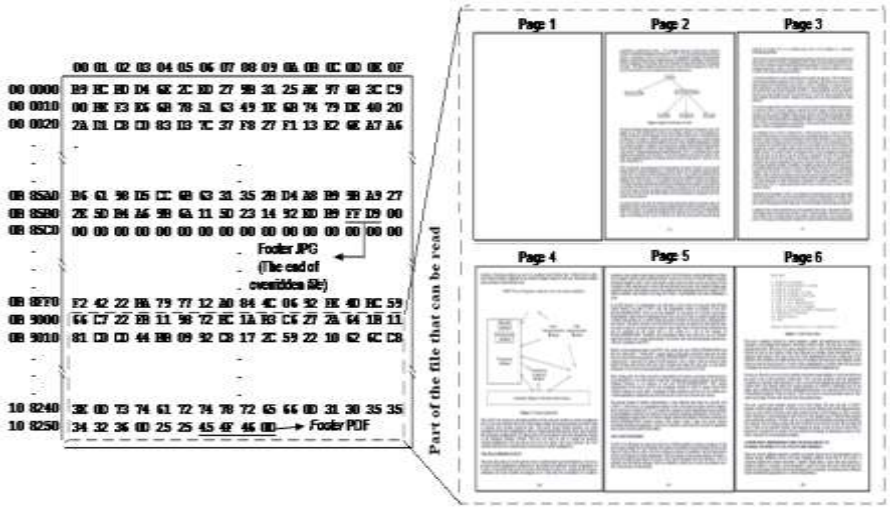

Figure 13. Some of readable pages from damaged PDF files

3.2. False Positive Analysis

False positive analysis is when the file is damaged or overwritten by another file but is still identified as a good file. This is possible if the file was overwritten by similar file smaller than the original file. Because the override file type was the same as the original file type, the header of the file which was found at the beginning of the file had the same signature as the original file. Therefore, if the identification results based on signature was compared with the result of

identification based on filename extension, then the file was identified as a good file. From the analysis, the results of false positive found were Microsoft documents (DOCX), PDF documents, and JPG image files.

Microsoft documents (DOCX) that were overwritten by other Microsoft documents could not be opened after they are restored even though the identification results indicate that the file was in good condition. When the document was checked, more than one signatures were found from the footer of the Microsoft documents. This fact indicates that the document had been overwritten by other Microsoft documents. Information of the Microsoft documents is shown in Table 7.

Table 7. Information of Microsoft documents (false positive)

Parameter	Value
File type based on *filename extension*	Microsoft Document (.DOCX)
File type based on *signature*	Microsoft Document (.DOCX)
Result of file damage test	Good File
File size	4,65 MB (4,884,115 bytes)
File size with slack space (file size in storage media)	4,66 MB (4,886,528 bytes)
Number of the footer signature found	2 on *offset*: • 0x010068 (50 4B 05 06 00 00 00 00 0E 00 0E 00 8D 03 00 00 DB FC 00 00 00 00) • 0x4A867D (50 4B 05 06 00 00 00 00 54 00 54 00 D3 16 00 00 AA 6F 4A 00 00 00)

The signature of the Microsoft document footer found on offset 0x4A867D (end of file) indicates that the footer signifies the end of the file. However, the footer found in the mid-file on offset 0x010068 indicates that it is the end of the file. Two footers found on two different offsets indicate that there had been a change in the actual file. Figure 14 illustrates the signature of Microsoft documents footer found on two different offsets.

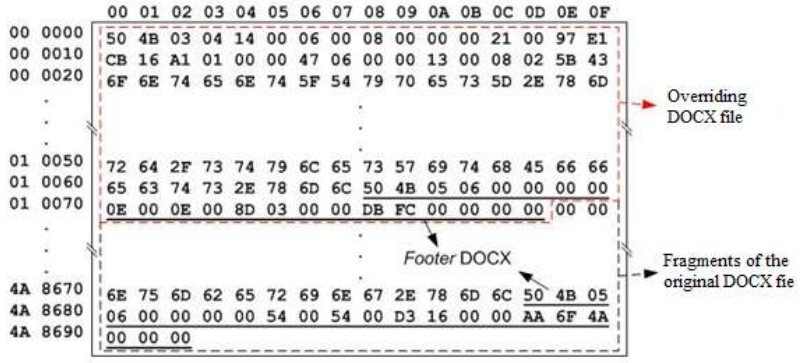

Figure 14. Signature of DOCX footer found on two offsets

The 4.66 MB document was overwritten by other documents with smaller size of 65,641 bytes (64.1 KB). This causes the some 68KB of the original document was overwritten so the actual data of the document was damaged. Because the overriding file type is the same as the initial file type, the files share the same signature on the header. Thus, the file was a corrupted file although it was identified as a good file (*false positive*).

Unlike Microsoft documents that can not be opened if some of the data in the file has been overwritten, JPG image and PDF files could still be opened even if it had been overwritten. This will cause the file to look as if it is still good because it can still be opened properly using the application. Some false positives could be detected by checking the file size where the file size is too large when compared to the file content. This can happen if the size difference between the override and the initial files is huge. For example, JPG file is a compressed image

file. When viewed from the width, height, resolution, and bit depth of the image, the file size is too large for a JPG image. There are two signatures of JPG image footer found in the image file on different offset of 0xD391 and 0xBA3EA. Information of JPG image file which is a false positive can be seen in Table 8.

Table 8. Information of JPG Image (False Positive)

Parameter	Value
File type based on *filename extension*	JPG file
File type based on *signature*	JPG file
Result of file damage test	Good file
Image width	800 pixel
Image height	800 pixel
Horizontal resolution	96 dpi
Vertical resolution	96 dpi
Bit depth	24
File size	744 KB (762,860 byte)
File size with slack space (file size in storage media)	744 KB (765,952 byte)
Number of the footer signature found	2 on offsets: a. 0xD391 (FF D9) b. 0xBA3EA (FF D9)

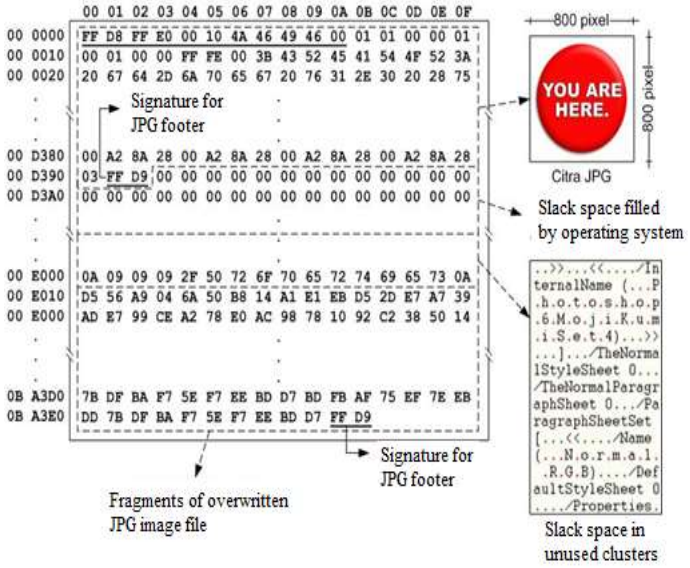

Figure 15. The search of JPG file content (Hexadecimal Values)

In Figure 15, the search of the file contents showed that the successfully opened JPG image with dimension of 800 pixel × 800 pixel was the image that overwrites the previous image. This image has a size of 52.8KB (56 KB if the slack space is included) in accordance with the footer signature found on the offset 0xD391. Thus, it is known that there was previously a JPG image file of 744 KB that was overwritten by a smaller JPG image. The override causes the first 56 KB of the initial JPG file are damaged so that the fragment of the initial JPG file with size of 688KB starts from offset 0xE011. The search of hexadecimal value of the file is shown in Figure 16.

Figure 16 shows the false positive of a PDF document. PDF file that become false positive have a content of 4 pages which consists mostly of text. However, the file size is too large at 1.39 MB (1,459,518 bytes). According to the search, the 4-page contents only occupy the first 56 KB (56,849 bytes) of the file which is up to offset 0x00DE15. It is also found within the file two PDF footers which located on *offset* 0x00DE0F with value of "0A 25 25 45 4F 46 0A" and at the end of the file on *offset* 0x164537 with value of "0D 25 25 45 4F 46 0D".

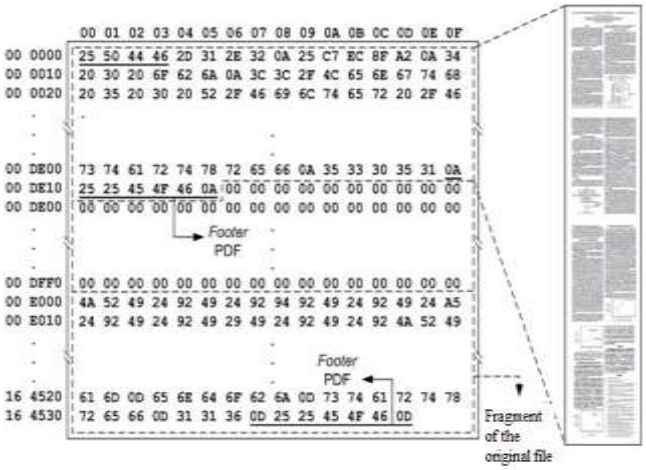

Figure 16. The search of PDF file content (hexadecimal values)

3.3. Testing Results

We conducted 4 testings with different data size in the media storage. The first testing shows that using 0 byte data size added to media storage and 56 files in MFT's entry, 55 files could be recovered (98,21%) and 1 file unsuccessfully recovered (1.79%) even though the file entry was still in MFT. Result of second testing was on the storage media with data overwritten by other data is 724,167,330 bytes or 18.98% from the storage media total size. Readable File entry on MFT is 55 files, 47 files are successfully recovered (83.92%), 8 files recovered with some damage, 1 file cannot be recovered, and the total damaged file size is 933,351,059 bytes. For the third testing with 55 files successfully read in MFT, 48 files unsuccessfully recovered, 7 damaged files, 1 failed to be recovered, and the total damaged file size is 1,631,210,540 bytes. Lastly, the forth testing shows 55 files entry on MFT, 46 were recovered, 9 recovered with damages, 1 file need to be recovered, and the damaged file size is 2,337,387,248 bytes. Results from the first to the fourth testing shown that the greater overwritten data size does not mean that the more files will be damage. On the contrary, the greater the size of a file can be recovered the more likely that the file will be damage at the time the overwritten is happened. The successful rate of undelete is depends on several factors, such as:

a. Condition of the MFT. The undelete process requires metadata obtained from MFT attribute so that if MFT is damage (for example the storage media is formatted), then the undelete process couldn't be performed.
b. The size of data overwritten on the storage media. The smaller the file size overwritten, the more successful the file recovery and vice versa.
c. The size of file to be recovered. The larger the file size, the more likely that file is overwritten by another data after the deletion and the successful recovery rate is also become smaller.

Based on the testing results from the first test until the fourth test, the average test results can be seen in Table 9. The average test result shows that 98.66% MFT entry can still be read after file deletion, 87.50% deleted files can still be recovered successfully, 10.71% files recovered with some damage, and 1.79% failed to be recovered.

Table 9. Average Test Values

Parameter	Average
Number of MFT entry read	98.66%
Number of file successfully recovered	87.50%
Number of file recovered with damage	10.71%
Number of file fail to recovered	1.79%

4. Conclusion and Future Work

In this paper, we have implemented a file undelete and the Aho-Corasick algorithms to reconstruct files in order to recover deleted files from a file system. The implementation of the proposed file undelete algorithm was able to recover 55 files with a total size of 3.52GB in 229.418 seconds, which count to average data processing speed of 15.77 MB/s. However, the proposed file reconstruction method was entirely depended on Master File Table (MFT) condition, meaning that if the MFT was damaged, it will affect the recovery result. In addition, the size of files to be recovered and the portion of overwritten file were also affecting the success rate of the recovery process.

String-matching method using Aho-Corasick algorithm implemented in the file type identification and damage checking was capable in finding the file signature and determining the damage by comparing the result of identification based on filename extension and signature. It should be noted that the identification of the damaged file could generate a false positive result if the file was overwritten by a similar file type. This would cause the signature found in the recovered file to be the same as the original file so that the file will be identified as a good file even if some of the file contents had been overwritten. In this research, string-matching method using Aho-Corasick algorithm was applied only to search signature of the first 32 bytes of the file in order to identify the file type.

For future research, we should be able to identify file types from file fragments, so the content of the fragments in a damaged file could be read by using the appropriate file signature. We could also conduct file header reconstruction of a damaged file so that the file could be read and reopened by suitable applications. Furthermore, we could also perform information extraction from a damaged file using a more efficient method so that all information in the damaged file could be recovered.

Acknowledgement

The authors would like to thanks Lembaga Penelitian Universitas Sumatera Utara in supporting this research work.

References

[1] Aburabie YT, Alomari M. Computer Forensic: Permanent Erasing. New York Institute of Technology (NYIT)-Jordan's campus. 2006.
[2] Lee TH. *Generalized Aho-Corasick Algorithm for Signature Based Anti-Virus Applications*. Proc. of 16th International Conference on Computer Communications and Networks (ICCCN 2007); 2007: 792-797.
[3] Richard III GG, Roussev V, Marziale L. In-Place File Carving. In: Craiger, P, Shenoi, S. (eds.) Advances in Digital Forensics III. IFIP. 242; 2007: 217–230.
[4] Kilpeläinen P. *Set Matching and Aho-Corasick Algorithm.* Biosequence Algorithm. Department of Computer Science. University of Kuoplo: Kuoplo. 2005.
[5] Bentley J, Sedgewick R. *Fast algorithms for sorting and searching strings*, Proc. ACM-SIAM Symposium on Discrete Algorithms. 1997: 360–369.
[6] Aygün RS. S2S: Structural-to-Syntactic Matching Similar Documents. *Knowledge and Information Systems*. 2008; 16(3): 303-329.
[7] Beebe NL, Clark JG. Digital Forensic Text String Searching: Improving Information Retrieval Effectiveness by Thematically Clustering Search Results. *Digital Investigation*. 2007; *4:* 49-54.
[8] Beebe N, Dietrich G. *A New Process Model for Text String Searching*. In: Craiger P., Shenoi S. (eds) Advances in Digital Forensics III. Digital Forensics 2007. IFIP-The International Federation for Information Processing. New York: Springer. 242: 179-191.
[9] Sajja A. Forensic Reconstruction of Variable Bitrates MP3 Files. Thesis. University of New Orleans: 2010.
[10] Yoo B, Park J, Lim S. A Study on A Multimedia Carving Method. *Multimedia Tools and Application*. 2012; 61(1): 243–261.
[11] Kurniawan F, Rahim MSM, Khalil MS, Khan MK. Statistical Based Audio Forensic on Identical Microphones. *International Journal of Electrical and Computer Engineering (IJECE)*. 2016; 6(5): 2211-2218.
[12] Gunawan TS, Hanafiah SAM, Kartiwi M, Ismail N, Za'bah NF, Nordin AN. Development of Photo Forensics Algorithm by Detecting Photoshop Manipulation Using Error Level Analysis. *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*. 2017; 7(1): 131–137.

[13] Kingra S, Aggarwal N, Singh RD. Video Inter-frame Forgery Detection Approach for Surveillance and Mobile Recorded Videos. *International Journal of Electrical and Computer Engineering (IJECE)*. 2017; 7(2): 831-841.
[14] Thanekar SA, Subrahmanyam K, Bagwan AB. A Study on Digital Forensics in Hadoop. *Indonesian Journal of Electrical Engineering and Computer Science*. 2016; 4(2): 473-478.
[15] Sitompul OS, Handoko A, Rahmat RF. *A File Undelete with Aho-Corasick Algorithm In File Recovery*. The International Conference on Informatics and Computing (lCIC). 2016: 427–431.
[16] Rahmat RF, Nicholas F, Purnamawati S, Sitompul OS. File Type Identification of File Fragments using Longest Common Subsequence (LCS). *Journal of Physics: Conference Series.* 2017; 801(1): 12054.
[17] Aaron, Sitompul OS, Rahmat RF. *Distributed autonomous Neuro-Gen Learning Engine for content-based document file type identification*. International Conference on Cyber and IT Service Management (CITSM), 2014: 63–68.
[18] Hasibuan ZA, Rahmat RF, Pasha MF, Budiarto R. Adaptive Nested Neural Network (ANNN) Based on Human Gene Regulatory Network (GRN) for Gene Knowledge Discovery Engine, *IJCSNS International Journal of Computer Science and Network Security.* 2009; 9(6): 43–54.
[19] Pasha MF, Rahmat RF, Budiarto R, Syukur M. A distributed autonomous neuro-gen learning engine and its application to the lattice analysis of cubic structure identification problem, *International Journal of Innovative Computing, Information and Control*. 2010; 6(3): 1005–1022.
[20] Rahmat R, Pasha M, Syukur M, Budiarto RA. Gene-Regulated Nested Neural Network, *International Arab Journal of Information Technology*. 2015; 12(6): 532–539.