International Journal of Internet, Broadcasting and Communication Vol.13 No.3 118-123 (2021)
http://dx.doi.org/10.7236/IJIBC.2021.13.3.118

IJIBC 21-3-15

File Signature's Automatic Calculation Algorithm Proposal for Digital Forensic

Eun-Jin Jang[1], Seung-Jung Shin[2]*

[1] Researcher, Department of IT Convergence, Hansei University, Korea
dmswls1061@naver.com
[2] Professor, Department of ICT Convergence, Hansei University, Korea*
expersin@hansei.ac.kr

Abstract

Recently, digital crime is becoming more intelligent, and efficient digital forensic techniques are required to collect evidence for this. In the case of important files related to crime, a specific person may intentionally delete the file. In such a situation, data recovery is a very important procedure that can prove criminal charges. Although there are various methods to recover deleted files, we focuses on the recovery technique using HxD editor. When recovering a deleted file using the HxD editor, check the file structure and access the file data area through calculation. However, there is a possibility that errors such as arithmetic errors may occur when a file approach through calculation is used. Therefore, in this paper, we propose an algorithm that automatically calculates the header and footer of a file after checking the file signature in the root directory for efficient file recovery. If the algorithm proposed in this paper is used, it is expected that the error rate of arithmetic errors in the file recovery process can be reduced.

Keywords: *Digital-Forensic, File Signature, Header and Footer, File Restoration, Hex Data*

1. Introduction

Today, the development of computer technology is breaking down online and offline barriers and increasing the number of online digital crimes. In this situation, collecting online evidence is more important than offline evidence to prove the crime, also this is one of the reason why digital forensic should be studied [1].

Although digital forensic techniques are very diverse, the technique of recovering deleted data among them is one of the most important techniques. When recovering a completely deleted file through the HxD editor, the structure of the file is checked and the data area of the file is accessed through calculation. However, there is a possibility that errors such as arithmetic errors may occur when a file approach through calculation is used. Therefore, in this paper, we propose an algorithm that automatically calculates the header and footer of a file after checking the file signature in the root directory for efficient file recovery.

Manuscript Received: July. 2, 2021 / Revised: July. 5, 2021 / Accepted: July. 8, 2021
Corresponding Author: expersin@hansei.ac.kr
Tel: +82-31-450-9585, Fax: +82-31-450-5172
Professor, Department of ICT Convergence, Hansei University, Korea

2. File System and File Signature

2.1 File System

Representatively used file systems include FAT32, NTFS, and exFAT. FAT32 (File Allocation Table32) is the most commonly used file system and has good compatibility, so it can be used in various operating systems and devices, and it is also safe. However, there is a limit to handling high-capacity files due to lack of scalability. NTFS (New Technology File System) is a format developed to compensate for the shortcomings of FAT32, and the maximum size of the drive is 256 TB. As such, the NTFS system is efficient in handling large files, but there is a limitation in its use in except for Windows other operating systems due to the lack of compatibility. Because NTFS is the Windows-oriented format. The exFAT (Extended File Allocation Table) format is a system developed to compensate for the shortcomings of NTFS, and has both extensibility and compatibility [2]. However, exFAT has a disadvantage of low stability. Table 1 shows the character of the main file systems.

Table 1. The character of File System

File System	Strength	Weakness
FAT32	compatibility, safety	scalability
NTFS	scalability	compatibility
exFAT	compatibility, scalability	safety

2.2 File Signature

Every file has its own format, and the most basic content of this format is called ‘File Signature’. File Signature has a unique value Header indicating the start of a file and a unique value Footer indicating the end of a file. Depending on the file format, only Header or both Header and Footer may exist [5]. Table 2 shows the file format in which both header and footer exist.

Table 2. File Format with both Header and Footer

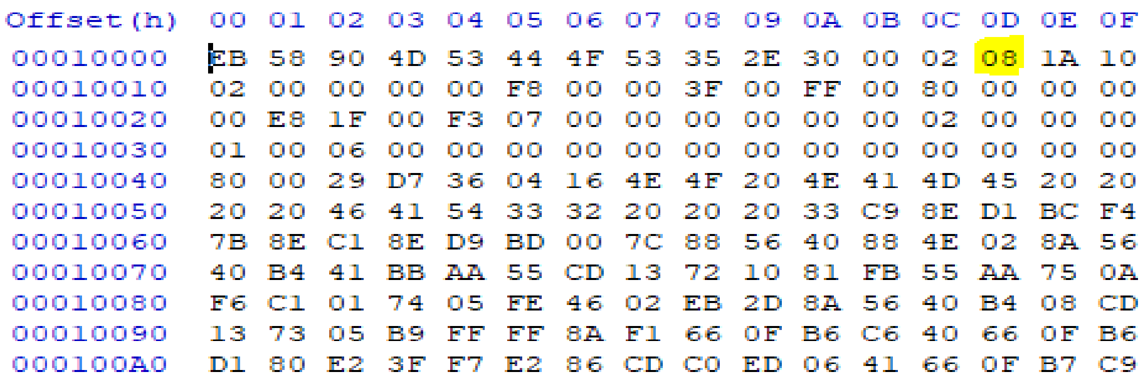
File Format	Header	Footer
xlsx, docx, pptx	50 4B 03 04 14 00 60 00	50 4B 05 06
PDF	25 50 44 46 2D 31 2E	25 25 45 4F 46
JPEG	FF D8	FF D9
GIF	47 49 46 38	00 3B
PNG	89 50 4E 47 0D 0A 1A 0A	49 45 4E 44 AE 42 60 82

3. The way of File Restoration using HxD

There are various methods to recover completely deleted files from PC, but we used the HxD editor for file recovery. Create a virtual drive for file recovery, set the partition format to MBR, and set the file system to FAT32.

3.1 File restoration sequence

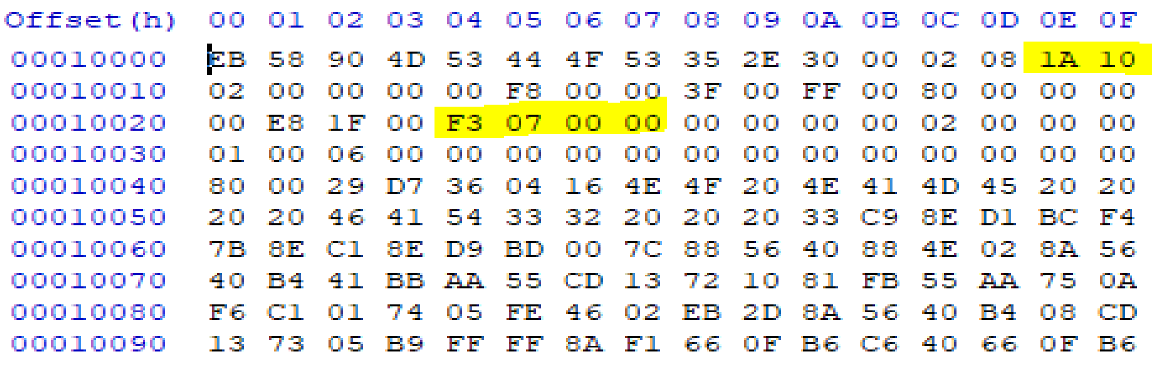
Since we used the MBR format in this paper, the boot code part is checked by checking the data structure of the MBR format. Since 0~445 is the Boot Code area, if you check the HEX value of 445, you can confirm that it is 18D. After 18D, 16 bytes correspond to one partition. Check the location of BR (Boot Record) by calculating LBAaddress of File Partition as Little-edian. In this paper, 00 00 00 80 is HEX 80, so BR becomes 128 sector. After moving to BR, checked the number of sectors per cluster. Figure 1 means the number of sectors per cluster identified in 128 sector locations.



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00010000   EB 58 90 4D 53 44 4F 53 35 2E 30 00 02 08 1A 10
00010010   02 00 00 00 00 F8 00 00 3F 00 FF 00 80 00 00 00
00010020   00 E8 1F 00 F3 07 00 00 00 00 00 00 02 00 00 00
00010030   01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00
00010040   80 00 29 D7 36 04 16 4E 4F 20 4E 41 4D 45 20 20
00010050   20 20 46 41 54 33 32 20 20 20 33 C9 8E D1 BC F4
00010060   7B 8E C1 8E D9 BD 00 7C 88 56 40 88 4E 02 8A 56
00010070   40 B4 41 BB AA 55 CD 13 72 10 81 FB 55 AA 75 0A
00010080   F6 C1 01 74 05 FE 46 02 EB 2D 8A 56 40 B4 08 CD
00010090   13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66 0F B6
000100A0   D1 80 E2 3F F7 E2 86 CD C0 ED 06 41 66 0F B7 C9
```

Figure 1. Number of Sectors per cluster

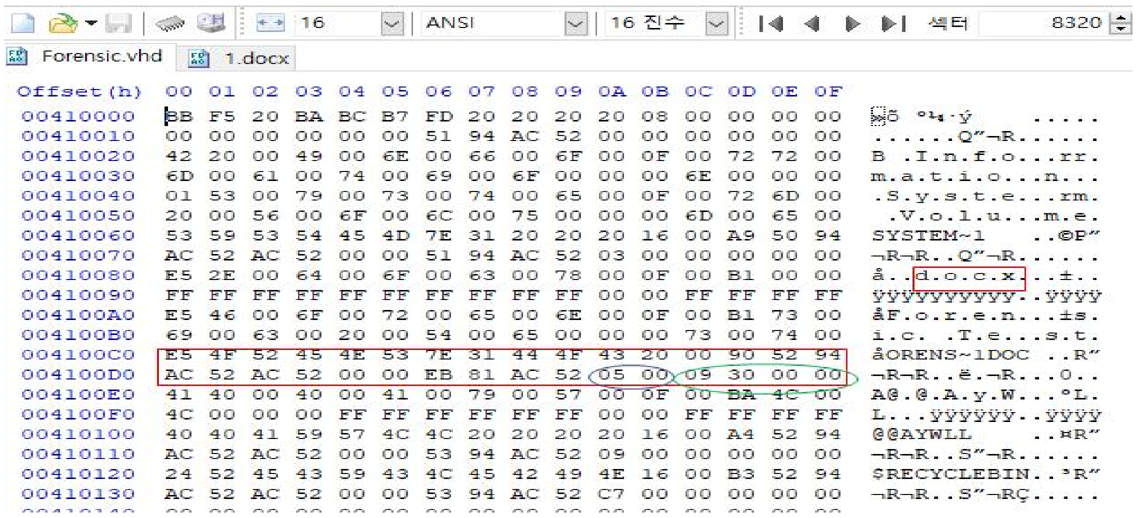
After checking the number of sectors per cluster, checked the size of Reserved Area and FAT32. As shown Figure 2, 1A and 10 means Reserved Area, and F3 07 00 00 means size of FAT32. Figure 2 shows the size of Reserved Area and FAT32.



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00010000   EB 58 90 4D 53 44 4F 53 35 2E 30 00 02 08 1A 10
00010010   02 00 00 00 00 F8 00 00 3F 00 FF 00 80 00 00 00
00010020   00 E8 1F 00 F3 07 00 00 00 00 00 00 02 00 00 00
00010030   01 00 06 00 00 00 00 00 00 00 00 00 00 00 00 00
00010040   80 00 29 D7 36 04 16 4E 4F 20 4E 41 4D 45 20 20
00010050   20 20 46 41 54 33 32 20 20 20 33 C9 8E D1 BC F4
00010060   7B 8E C1 8E D9 BD 00 7C 88 56 40 88 4E 02 8A 56
00010070   40 B4 41 BB AA 55 CD 13 72 10 81 FB 55 AA 75 0A
00010080   F6 C1 01 74 05 FE 46 02 EB 2D 8A 56 40 B4 08 CD
00010090   13 73 05 B9 FF FF 8A F1 66 0F B6 C6 40 66 0F B6
```

Figure 2. Reserved Area and size of FAT32

We can check the value of the Root Directory sector by adding the value obtained by adding the Reserved Area and FAT32 twice to BR. Going to the Root Directory location and check the format of the file. Then, after checking the location and size of the file, the file is restored. Figure 3 shows the file format, location and size checked in the Root Directory, and Figure 4 shows the restored file.



```
Offset(h)  00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00410000   BB F5 20 BA BC B7 FD 20 20 20 20 08 00 00 00 00  »õ º¼·ý    .....
00410010   00 00 00 00 00 00 51 94 AC 52 00 00 00 00 00 00  ......Q”¬R......
00410020   42 20 00 49 00 6E 00 66 00 6F 00 0F 00 72 72 00  B .I.n.f.o...rr.
00410030   6D 00 61 00 74 00 69 00 6F 00 00 00 6E 00 00 00  m.a.t.i.o...n...
00410040   01 53 00 79 00 73 00 74 00 65 00 0F 00 72 6D 00  .S.y.s.t.e...rm.
00410050   20 00 56 00 6F 00 6C 00 75 00 00 00 6D 00 65 00   .V.o.l.u...m.e.
00410060   53 59 53 54 45 4D 7E 31 20 20 20 16 00 A9 50 94  SYSTEM~1   ..©P”
00410070   AC 52 AC 52 00 00 51 94 AC 52 03 00 00 00 00 00  ¬R¬R..Q”¬R......
00410080   E5 2E 00 64 00 6F 00 63 00 78 00 0F 00 B1 00 00  å..d.o.c.x...±..
00410090   FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿ..ÿÿÿÿ
004100A0   E5 46 00 6F 00 72 00 65 00 6E 00 0F 00 B1 73 00  åF.o.r.e.n...±s.
004100B0   69 00 63 00 20 00 54 00 65 00 00 00 73 00 74 00  i.c. .T.e...s.t.
004100C0   E5 4F 52 45 4E 53 7E 31 44 4F 43 20 00 90 52 94  åORENS~1DOC ..R”
004100D0   AC 52 AC 52 00 00 EB 81 AC 52 05 00 09 30 00 00  ¬R¬R..ë.¬R...0..
004100E0   41 40 00 40 00 41 00 79 00 57 00 0F 00 BA 4C 00  A@.@.A.y.W...ºL.
004100F0   4C 00 00 00 FF FF FF FF FF FF 00 00 FF FF FF FF  L...ÿÿÿÿÿÿ..ÿÿÿÿ
00410100   40 40 41 59 57 4C 4C 20 20 20 20 16 00 A4 52 94  @@AYWLL    ..¤R”
00410110   AC 52 AC 52 00 00 53 94 AC 52 09 00 00 00 00 00  ¬R¬R..S”¬R......
00410120   24 52 45 43 59 43 4C 45 42 49 4E 16 00 B3 52 94  $RECYCLEBIN..³R”
00410130   AC 52 AC 52 00 00 53 94 AC 52 C7 00 00 00 00 00  ¬R¬R..S”¬RÇ.....
```

Figure 3. File Format, Location and Size

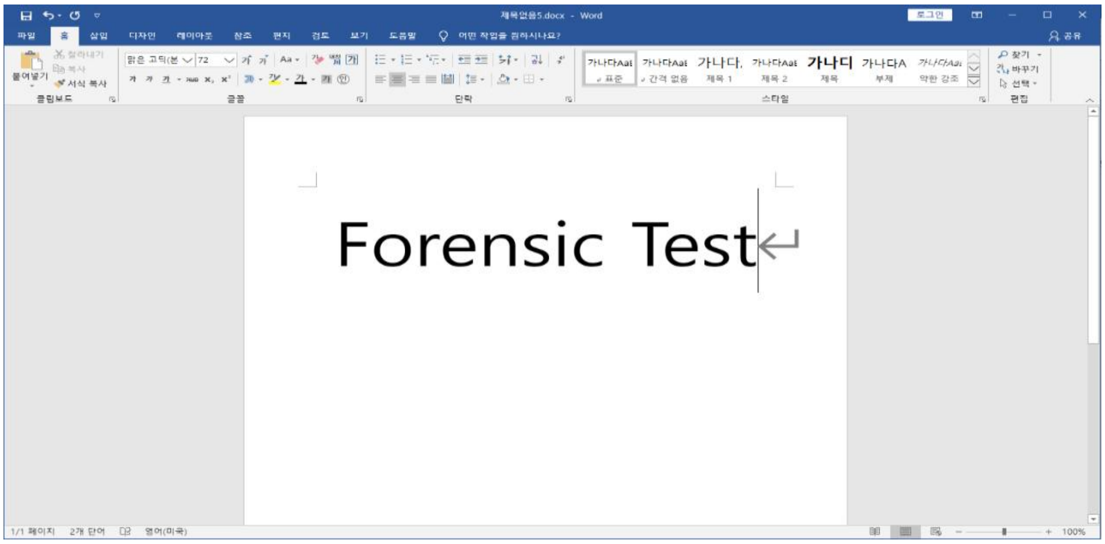


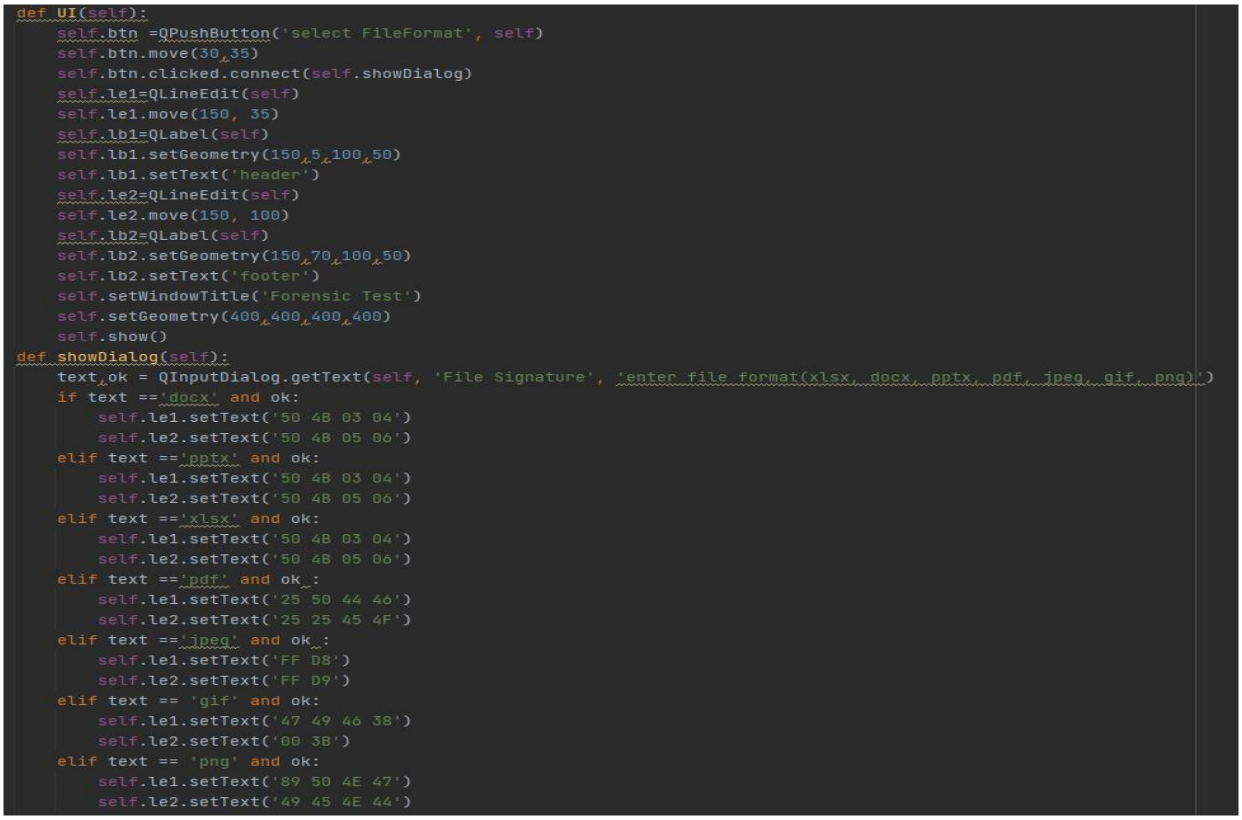
Figure 4. Recovered File

In this way, the deleted file can be recovered using the HeX editor, but errors such as operation errors can occur because the data area can be accessed through a series of calculations. Therefore, this paper proposes an algorithm that automatically calculates the header and footer of a file after checking the file signature in the root directory for efficient file recovery.

4. Proposed Automatic Calculation Algorithm

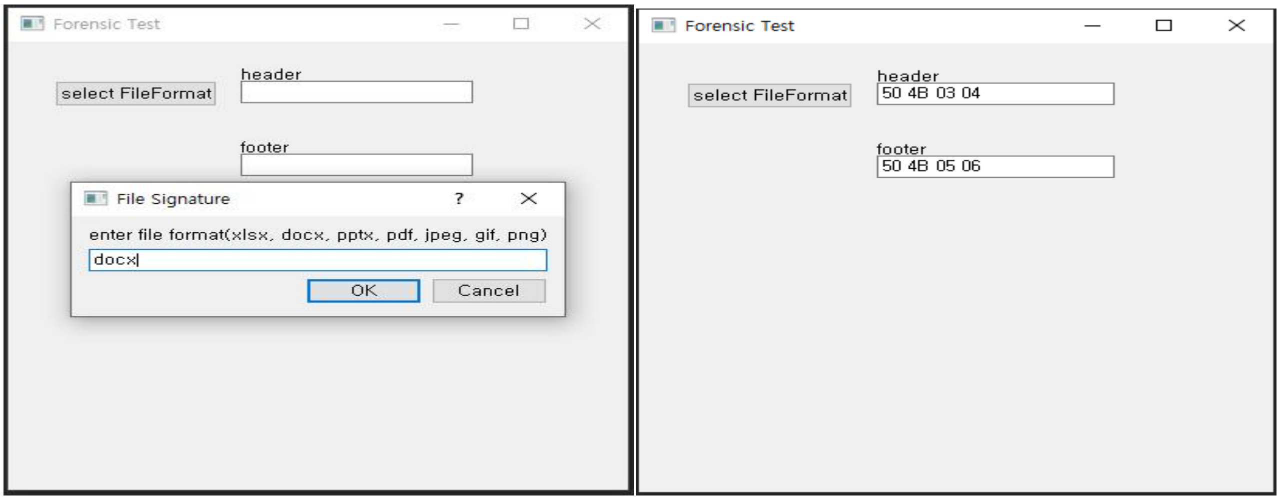
EnCase, an integrated digital forensic tool, also has a function to check the signature of a file. Due to the nature of the forensic tool that comprehensively checks the file structure, if the signature is damaged or the extension is changed, it is difficult to confirm the clear result value. However, in the case of the automatic calculation system proposed in this paper, there is a difference in that a clear result value can be confirmed because the header and footer are checked by inputting the file format checked in the Root Directory.

Since the algorithm proposed in this paper is the algorithm that after checking the file type when the file type is input, calculates the header and footer. So this algorithm is aim for running file forms including header and footer as shown in Table 2 As the development environment, the OS is Window64 bit, the development language is Python 3.9, and the development tool is PyCharm. Figure 5 shows the code of the proposed algorithm, and Figure 6 shows the calculation algorithm execution screen.



```python
def UI(self):
    self.btn =QPushButton('select FileFormat', self)
    self.btn.move(30,35)
    self.btn.clicked.connect(self.showDialog)
    self.le1=QLineEdit(self)
    self.le1.move(150, 35)
    self.lb1=QLabel(self)
    self.lb1.setGeometry(150,5,100,50)
    self.lb1.setText('header')
    self.le2=QLineEdit(self)
    self.le2.move(150, 100)
    self.lb2=QLabel(self)
    self.lb2.setGeometry(150,70,100,50)
    self.lb2.setText('footer')
    self.setWindowTitle('Forensic Test')
    self.setGeometry(400,400,400,400)
    self.show()
def showDialog(self):
    text,ok = QInputDialog.getText(self, 'File Signature', 'enter file format(xlsx, docx, pptx, pdf, jpeg, gif, png)')
    if text =='docx' and ok:
        self.le1.setText('50 4B 03 04')
        self.le2.setText('50 4B 05 06')
    elif text =='pptx' and ok:
        self.le1.setText('50 4B 03 04')
        self.le2.setText('50 4B 05 06')
    elif text =='xlsx' and ok:
        self.le1.setText('50 4B 03 04')
        self.le2.setText('50 4B 05 06')
    elif text =='pdf' and ok :
        self.le1.setText('25 50 44 46')
        self.le2.setText('25 25 45 4F')
    elif text =='jpeg' and ok :
        self.le1.setText('FF D8')
        self.le2.setText('FF D9')
    elif text == 'gif' and ok:
        self.le1.setText('47 49 46 38')
        self.le2.setText('00 3B')
    elif text == 'png' and ok:
        self.le1.setText('89 50 4E 47')
        self.le2.setText('49 45 4E 44')
```

Figure 5. Code of Calculating Algorithm



Forensic Test

header

footer

File Signature

enter file format(xlsx, docx, pptx, pdf, jpeg, gif, png)

docx

Forensic Test

header 50 4B 03 04

footer 50 4B 05 06

Figure 6. Algorithm Run Screen

5. Conclusions

In this paper, we proposed an algorithm that automatically calculates headers and footers by analyzing file signatures by file type as a way to recover deleted file data. The proposed algorithm targets the files to be restored using the HxD editor. When recovering files using the HxD editor, it is essential to perform file-type specific calculations. However, when deleted files were recovered by a simple calculation, data recovery safety such as operation errors and functional errors may be deteriorated. Using the algorithm proposed in this paper, it is possible to check the start and end points of a file without a separate calculation, and it is expected that more stable file deletion recovery is possible by lowering the error rate of operation errors. In the future, if more various file formats are supported in addition to the file formats tested in this algorithm, also additional functions and UI for algorithm testing are reorganized, it is expected that more diverse studies on deleted file recovery will proceed.

References

[1] Gyusang Cho, “A Maximum Data Allocation Rule for an Anti-forensic Data Hiding Method in NTFS Index Record”, The International Journal of Internet, Broadcasting and Communication(IJIBC), Vol. 9, No. 3, pp. 17-26, May 2017.
DOI: https://doi.org/10.7236/IJIBC.2017.9.3.17

[2] Jiyoon Ham, Joshua I. James, “A Feature Comparison of Modern Digital Forensic Imaging Software”, The Journal of The Institute of Internet, Broadcasting and Communication (IIBC), Vol. 19, No. 6, pp. 15-20, Dec 2019
DOI: https://doi.org/10.7236/JIIBC.2019.19.6.15

[3] Eunjin Jang, Seungjung Shin, “Proposal of New Data Processing Function to Improve the Security of Self-driving Cars System”, The Journal of The Institute of Internet, Broadcasting and Communication(JIIBC), Vol.20, No.4, pp.81-86, Aug 2020
DOI: http://doi.org/10.7236/JIIBC.2020.20.4.81

[4] Dohyun Kim, Junki Kim, Sangjin Lee, “An Analysis of Google Cloud Data from a Digital Forensic Perspective”, The Journal of The Korea Institute of Information and Communication Engineering(JKIICE), Vol.24, No.12, pp.1662-1669, Dec 2020
DOI: http://doi.org/10.6109/JKIICE.2020.24.12.1662

[5] Yeonjoo Lee, Jeongmin Kim, Sungjin Lee, “A Study of Polaris Office Forensic Artifact”, The Journal of Digital Forensics(KDFS), Vol.14, No.4, pp.368-378, Dec 2020
DOI: http://doi.org/10.22798/KDFS.2020.14.4.368

[6] Soojin Kang, Sumin Shin, Giyoon Kim, Jongsung Kim, “Forensic Analysis of Locking Pictures, Hiding Photos/Video and Safe Gallery/Camera Applications”, The Journal of Digital Forensics(KDFS), Vol.14, No.2, pp.125-138, Jun 2020
DOI: http://doi.org/10.22798/KDFS.2020.14.2.125

[7] Seyool Park, Sangjin Lee, “Forensic Investigation of HWP File”, The Journal of Digital Forensics(KDFS), Vol.14, No.4, pp.408-425, Dec 2020
DOI: http://doi.org/10.22798/KDFS.2020.14.4.408

[8] Woohwan Nam, “Android Emulators Forensic Analysis Technique”, The Journal of Digital Forensics(KDFS), Vol.13, No.4, pp.303-315, Dec 2019
DOI: http://doi.org/10.22798/KDFS.2019.13.4.303

[9] Seunghee Seo, Yeog Kim, Changhoon Lee, “Countering Portable Executable File Header Removing based Anti-Memory Forensic Techniques”, The Journal of Digital Forensics(KDFS), Vol.15, No.2, pp.50-59, Jun 2021
DOI: http://doi.org/10.22798/KDFS.2021.15.2.50

[10] Sanghuk Yoon, Sangjin Lee, “A Study on Digital Evidence Automatic Screening System”, The Journal of Digital Forensics(KDFS), Vol.14, No.3, pp.239-251, Sep 2020
DOI: http://doi.org/10.22798/KDFS.2020.14.3.239