# Identification of data wiping tools based on deletion patterns in ReFS $Logfile


By:
Eun Ji Lee, Seo Yeon Lee, Hyeon Kwon, Sung Jin Lee, Gi Bum Kim



From the proceedings of
The Digital Forensic Research Conference
**DFRWS APAC 2023**
Oct 17-20, 2023

Forensic Science International: Digital Investigation 46 (2023) 301607

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

# Identification of data wiping tools based on deletion patterns in ReFS $Logfile

Eun Ji Lee [a], Seo Yeon Lee [a], Hyeon Kwon [b], Sung Jin Lee [a], Gi Bum Kim [a,*]

[a] Dept. of Forensic Sciences, Sungkyunkwan University, 25-2 Sungkyunkwan-Ro, Jongno-Gu, Seoul, 03063, South Korea
[b] Deloitte Anjin LLC, Yeongdeungpo-gu, Seoul, South Korea

A R T I C L E I N F O

*Keywords:*
Data wiping
File deletion
ReFS logfile
Opcode
Anti-forensics
Digital forensics

A B S T R A C T

Data wiping tools permanently delete files by repeatedly overwriting data on a digital device, making file recovery impossible. Unlike the conventional deletion methods, which merely remove the file system pointer to the data, these tools are designed to entirely and irretrievably erase the data. This method can be exploited to obliterate evidence of a crime. Given the growing prevalence of such tools, a comprehensive analysis of permanent deletion behavior is essential, especially concerning the Resilient File System (ReFS). In this study, we propose a method for detecting user behavior concerning data wiping tools and algorithms in ReFS 3.7. Our approach relies on the fact that file modifications are logged in the redo record of the $Logfile, and that the opcode value of the redo record varies depending on the data wiping tool used. Since opcodes were only analyzed up to version 3.4, we analyzed the newly updated opcodes. Initially, we selected the 12 most commonly used data wiping tools for our research. In the pattern analysis phase, we applied the algorithms supported by each tool, generating a distinct deletion pattern for each one. This was accomplished by utilizing consecutive opcodes to formulate the patterns and monitor transitions in file and directory names. The patterns discerned in the $Logfile allowed us to determine which data wiping tool was deployed. The proposed methodology simplifies the identification of not only which data wiping tool has been used, but also the specific deletion behavior exhibited. We developed a tool incorporating the proposed method. Our subsequent verification confirmed the effectiveness of our methodology and tools in accurately detecting the use of comprehensive deletion tools. These findings contribute valuable insights to the acquisition of digital evidence of user deletion behavior in ReFS. Our proposed methodology will help digital forensic examiners in the detection and identification of data wiping tools' behavior.

## 1. Introduction

ReFS is a Windows filesystem developed by Microsoft since the launch of NTFS the intention of maximizing data availability, ensuring data integrity, and providing resilience against corruption. Compared to NTFS supporting 256 TB, ReFS vastly expands available space with support for an endorsed size of 35 PB (Microsoft, 2023). Despite ReFS being less commonly used than other file systems its usage doesn't exempt it from potential criminal activities (Brinkmann, 2023). As a non-bootable system, ReFS has fewer artifacts compared to others. Additionally, digital forensic tools such as Encase, Axiom, FTK, and Autopsy have limitations when dealing with ReFS. They only analyze up to specific versions or do not normally work, so that makes it difficult to find the user behavior, unlike other file systems. Therefore, it's essential to develop forensic analysis research to acquire evidence related to criminal activities. In this study, we focus on the traces left by the usage of data wiping tools in illicit activities. Internationally, there have been numerous criminal cases related to data wiping tools, such as the Virginia case (Woolwine, 2022) and the Swansea case (Jason Evans, 2022). Politically controversial cases have also emerged, like the one involving illegal surveillance of civilians by the South Korean Prime Minister's Office in June 2008. The court accepted the evidence of data wiping tool usage from an external computer as proof of document concealment (Lee, 2020). However, it could only be considered evidence when it was proven that the wiping tool used was identical to the program installed on the PC.

DFRWS 2023 APAC - Proceedings of the Third Annual DFRWS APAC
* Corresponding author.
*E-mail address:* freekgb02@gmail.com (G.B. Kim).

https://doi.org/10.1016/j.fsidi.2023.301607

Available online 13 October 2023

This paper proposes a methodology for identifying evidence of specific wiping tool behavior through an opcode pattern analysis of the $Logfile. To provide insights into complete deletion tools, we analyzed the functionalities and algorithms of the top 12 widely used tools, including Easy File Shredder and Kernel File Shredder. A meticulous examination was conducted to scrutinize the distinct patterns and relevant information that manifests in $Logfile throughout the course of data wiping using a range of deletion tools. These data wiping tools utilize their own algorithms for overwrite files multiple times. The deletion is executed by applying algorithms supported by each wiping tool, including the Gutmann algorithm (Gutmann, 1996a), a deletion method that overwrites data 35 times. We conducted experiments to investigate whether there are variations in patterns based on the deletion algorithm. We raised the question of whether pattern alterations depend on the use of a specific algorithm, as well as on the tools themselves. Furthermore, we analyzed filename changes to determine whether the deletion behavior can be traced back to specific characteristics.

The paper is structured as follows: Section 2 reviews previous studies on ReFS and deletion tools. Section 3 presents the environmental settings in which the methodology was applied and the corresponding results. Section 4 details the implementation of the method and the validation of the results obtained from Section 3. Section 5 concludes with remarks on our findings.

## 2. Related research

### 2.1. Concept of ReFS

The key attributes of ReFS encompass integrity streams, storage space integration, data recovery, proactive error correction, real-time tier optimization, and VM acceleration optimization. Research had suggested that the data availability and resilience offered by ReFS could see it become a more commonly used file system in the future (Lee et al., 2021). It supports block cloning, sparse VDL, file-level snapshots, and real-time tier optimization, which are features not present in NTFS. Moreover, structurally, it uses B + trees, a uniform disk structure that represents all disk-based information (Gudadhe et al., 2015).

ReFS is a journaling file system that maintains journal files like the Change Journal and $Logfile to document actions and modifications (Savoldi et al., 2012). While the Change Journal records the change history, including file and directory names, it doesn't allow for the extraction of sequenced behavior patterns with files. Therefore, in this study, we focused on the $Logfile, aiming to identify transactions based on opcode. $Logfile can be used as a forensic artifact by analyzing metadata related to work activities and target files, hence it was chosen for our research.

### 2.2. Wiping tools and erasure algorithms

Research on wiping activities was conducted on various Windows artifacts analysis of file systems. Savoldi et al. (2012) discussed scenarios related to data deletion cases and introduced a methodology to detect artifacts on a disk. They also used statistical methods to identify deleted regions on disks filled with random but periodic rule data. A different approach to our study on $Logfile in ReFS was proposed by Kim and Lee (2015), which aimed to identify the data wiping tool by extracting the trace time with the Amcache.hve. Shin et al. (2016) analyzed the file structure and log storage for a damaged EVTX(Windows XML Event Log) and proposed a recovery method for Chunk and Record units. Their comparison and analysis of both recovery methods suggested that the Chunk unit recovery method is more efficient if there hasn't been intentional damage, otherwise, the Record method is more effective. Smith et al. (2017) documented forensic artifacts that can collect and recover digital evidence from VMDK files. Their research centered around identifying forensic artifacts and their locations in virtualized computing to provide foundational knowledge for future digital forensic investigations. Cho (2018) addressed the problem of existing data hiding methods by bit correction in Timestamp of $MFT. There have been numerous studies analyzing journal files such as $Logfile, where transaction operations are recorded, and $UsnJrnl, which records changes in files and directories. Kim et al. (2020) analyzed user behavior in $UsnJrnl files and proposed a method to recover deleted $UsnJrnl files. They recovered $J property records of deleted journal files in non-allocation areas through file carving and found that they could recover a minimum of 75 to a maximum of 39,912 compared to $UsnJrnl:$J files obtained in live areas. Oh et al. (2021) discussed NTFS Log Tracker v1.7, which is used to manipulate the installation, execution, and use of suspicious tools based on signatures and patterns. Prior studies have also conducted experiments using specific data wiping tools. Jones and Afrifa (2020) performed an experiment on the efficacy of 8 data wiping tools and discovered that data remained when using Super File Shredder and Disk wipe. Horsman (2021) analyzed changes and logs of file properties with 8 data wiping tools in NTFS and FAT, demonstrating the characteristics that appeared when using data wiping tools. Unlike NTFS, which records both Redo and Undo opcodes, ReFS exclusively records Redo opcodes and has a unique filesystem structure. AlHarbi et al. (2022) confirmed that all files were successfully deleted and non-recoverable with 4 data wiping tools. They analyzed the file name that remained in the metadata after file deletion, presented the characteristics of each tool, and showed that the data wiping tool did not alter the internal information of the journal file. However, none of these studies targeted ReFS, and Horsman only viewed fragmentary records remaining in the log using existing log tools. Data wiping tools are generally count-based algorithms developed for confidentiality, such as U.S. DoD 5220, and so forth. These data wiping tools include the erasure algorithms listed in Table 1. The algorithms differ in the number of passes and speed. They also vary in the data and methods for overwriting. E.g., the Gutmann algorithm overwrites the target location 35 times with specific hexadecimal data, such as 00, ff, complement, or random values, while the other algorithms generally overwrite with decimal data between one to seven times.

### 2.3. ReFS forensics

Research on ReFS forensics has been conducted with a limited scope, primarily focusing on structural analysis and internal operational principles. Prade et al. (2019) applied digital forensic methods to version 3.4 of ReFS, analyzing changes in data when utilizing carving technology.

**Table 1**
Global erasure algorithm standard.

| No | Algorithm | Pass | Speed |
|---|---|---|---|
| 1 | HMG IS5 Base Line(British) (Jones and Afrifa, 2020) | 1 | Fast |
| 2 | GOST R 50739–95(Russian) (Russia, 1995) | 2 | Fast |
| 3 | Air Force(AFSSI) 5020(US) (U.S. Air Force, 1998) | 3 | Fast |
| 4 | Army AR380-19(US) (U.S. Army, 1998) | 3 | Middle |
| 5 | DoD 5220.22 M (US) (U.S. Defense Security Services, 2007) | 3 | Slow |
| 6 | DoD 5220.22 M E(US) (U.S. Defense Security Services, 2007) | 3 | Fast |
| 7 | DoE M-205.1-2 (US) (U.S. Dept. of energy, 2005) | 3 | – |
| 8 | HMG IS5 Enhanced (Jones and Afrifa, 2020) | 3 | Fast |
| 9 | ITSG2006 (Easy file shredder) | 3 | Fast |
| 10 | NAVSO P-5239-26(MFM) (US Navy) (U.S. Dept) | 3 | Fast |
| 11 | NAVSO P-5239-26(RLL) (US Navy) (U.S. Dept) | 3 | Fast |
| 12 | DoD 5220.28 STD (WASHINGTON, 1978) | 7 | Slow |
| 13 | DoD 5220.22 M E C E (WASHINGTON, 1978) | 7 | Middle |
| 14 | Bruce Schneier (Schneier, 1995) | 7 | Middle |
| 15 | VSITR (Bundesamts fr Sicherheit in der, 2004) | 7 | Slow |
| 16 | RCMP TSSIT OPS-II (RCMP G2-003, 2003) | 7 | Middle |
| 17 | N.A.T.O (Bitraser) | 7 | – |
| 18 | Peter Gutmann (Gutmann, 1996a) | 35 | Very Slow |

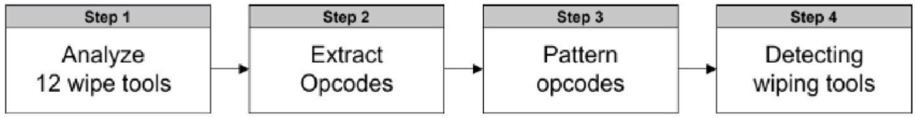



**Fig. 1.** Analysis methodology.

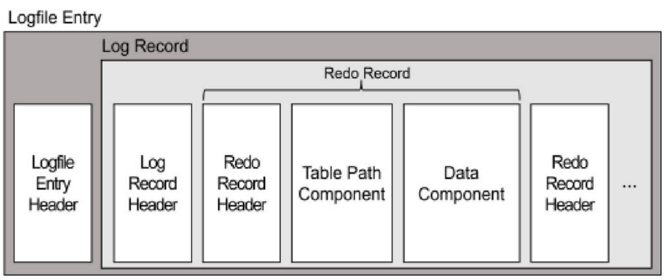



**Fig. 2.** Logfile entry structure.

However, this approach needs to be improved in its ability to discern user behavior as it concentrates solely on recovery. Kim (2019) proposed a method to recover deleted data by analyzing ReFS's internal metadata. The author compared the characteristics of existing large file systems, analyzed data storage principles, and presented two recovery algorithms. Nordvik et al. (2019) demonstrated that tracking deleted files was possible by analyzing the File Name Attribute (FNA) and data attributes using reverse engineering, yet made no attempt to analyze user behavior. Lee et al. (2019, 2021) began an opcode analysis of the $Logfile on ReFS version 3.4 to discern patterns of user behavior. Their studies on ReFS have primarily analyzed user behavior, but not focused on wiping. These studies only considered file creation, renaming, content modification, and normal deletion.

## 3. Experiment and result

### 3.1. Methodology

#### 3.1.1. Experiment methodology

To analyze the $Logfile generated after using data wiping tools, we used VMware Workstation 16.2.2 and Windows Server 2022 Standard Evaluation (x64-based, 21H2, OS build 20348.587) machine running on Intel Core i7-6700K processor with 32 GB DIMM 2133 MHz RAM. We created a partition on Windows and formatted the drive as ReFS(version 3.7) with size of 20,000 MB. We also considered SSDs due to their different characteristics from hard disks (Maneas et al., 2021). To verify if methods applied to HDDs can be applied to SSD (Samsung T7 1 TB), we used the same virtual machine environment. We formatted the external disk as ReFS and performed the same experiment by connecting a Portable SSD formatted with ReFS. We could obtained the same pattern results when inspecting the log file on the SSD as we did with the HDD.

After creating files with the source script (Matuzalem, 2022), and directories using the File Explorer 'New context menu', we wiped them using both the built-in delete function and data wiping tools. Also, it was executed on the files using the algorithms supported by each tool. We selected tools that operated the delete function properly in Windows ReFS and were publicly available data wiping tools. These tools are freely available for download on the internet, offer various deletion functions, and are easily accessible to users.

The analysis outlines the process of generating opcode patterns to discern user behavior with a data wiping tool. Fig. 1 shows a flow of the analysis methodology suggested in this study. In the first step, we analyzed 12 wiping tools to manage user behavior regarding the deletion function. After formatting the disk with ReFS, we created several files and directories then deleted the files using the tools with their algorithms. In the second step, we extract the opcodes from the $Logfile. In the third step, the opcode patterns derived from the same task were integrated. These derived patterns were categorized into 12 tools. Lastly, we applied the structure analysis method to the data wiping tools to derive File Creation patterns and automated this process for the development of a program that detects the execution of wiping tools.

**Table 2**
Opcodes and operations.

| Opcode | Redo operation | Version |
|---|---|---|
| 0x01 | Redo Insert Row | 3.4 (Lee et al., 2021), 3.7 |
| 0x02 | Redo Delete Row | |
| 0x04 | Redo Update Data with Root | |
| 0x05 | Redo Reparent Table | |
| 0x06 | Redo Allocate | |
| 0x07 | Redo Free | |
| 0x08, 0x09 | Redo Set Range State | |
| 0x10 | Redo Value as Key | |
| 0x12 | Copy Key Helper | |
| 0x0F | Redo Delete Table | |
| 0x1D | Redo Unlink Parent ID | 3.7 |
| 0x1E | Redo Value as Longlong | |
| 0x1F | Redo Update Stream Summary | |
| 0x20 | Redo Value as Key | |

#### 3.1.2. Logfile analysis methodology

The data area of the Logfile contains the $Logfile entry, which is composed of a Header and Log record, as shown in Fig. 2. Each Log record consists of multiple different redo records, including a Header, Data Offset Array, and Transaction Data, with an offset size of 1000. The Redo record commences at position 0xB0. The Transaction Data is comprised of a Table path and Data (Prade et al., 2019), allowing us to discern metadata such as the filename and date. Redo records are utilized to reconstruct changes in the event of a file system error in the Data Component (Russinovich, 2012). When a change occurs in the file system, information about the change and its location is stored to be used for recovery (Lee et al., 2021). Changes that have not yet been committed can be reconstructed. Opcodes are recorded in the redo record, reflecting internal changes prompted by user behaviors in the file. A previous study (Lee et al., 2021) identified 28 opcodes for ReFS redo operations in version 3.4. Opcodes were only discovered up to '1C' because that's all that existed in version 3.4. However, we identified the newly emerged '1D', '1E', and '1F' in version 3.7. by analyzing the refs.sys file using the IDA free 8.3.230608. Table 2 shows some opcodes

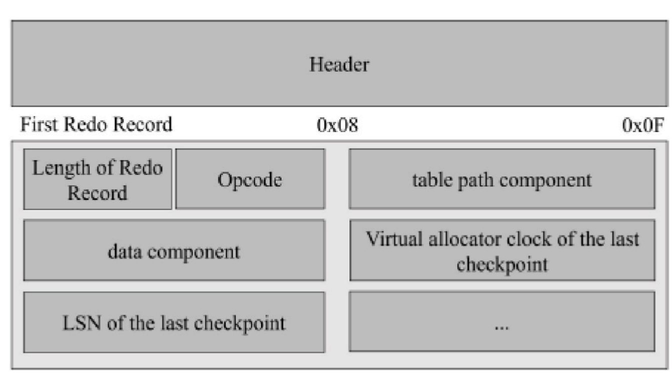



**Fig. 3.** Detail of redo record (Prade, 2019).

**Table 3**
Wiping tool list and algorithms (O: Available, -: Not Available, Δ: Available only full version).

| Tool Name | Easy File Shredder | File Shredder | Hard Wipe | Kernel File Shredder | PC Shredder | Remo File Eraser | Secure Eraser | Super File Shredder | Turbo Shredder | Wipe File | xShredder | XT File Shredder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Version | 2.0.2022 (U.S. Defense Security Services, 2007) | 2.5 (FileShredder) | 5.2.1 (Hardwipe) | 11.04.0 (Kernel file shredder) | 1.1 (PCShredder, 2008) | 2.0.0.5 (Remo Software) | 1.0.0 (Secure Eraser) | 4.12 (Kakasoft) | 0.036 (Turbo Shredder) | 3.6 (Wipe File) | 7.7.4.9 (xShredder) | 2.1 (XT File Shredder) |
| One(Simple) | – | O | – | – | – | – | – | O | O | O | – | O |
| Secure | – | O | – | – | – | – | – | O | O | O | – | – |
| Random | O | – | O | – | – | O | O | – | O | O | – | – |
| Zero | O | – | O | O | – | O | – | – | O | O | – | – |
| HMG IS5 BaseLine& Enhanced | – | – | – | O | – | – | – | – | – | – | O | – |
| GOST P50739-95 | O | – | O | O | – | – | – | – | – | – | O | – |
| Air Force 5020 | O | – | – | O | – | – | – | – | – | – | O | – |
| AR380-19 | O | – | – | O | – | – | – | – | – | – | O | – |
| DoD 5220.M 3 | O | O | O | O | O | O | – | O | – | – | O | O |
| DoD 5220.M E | O | – | – | – | – | – | O | – | – | – | – | O |
| ITSG2006 | O | – | – | – | – | – | – | – | – | – | – | – |
| NAVSO P-5239-26(RLL) | – | – | – | – | – | Δ | – | – | – | – | – | – |
| DoD 5220.28 STD | O | – | – | – | – | Δ | – | – | – | – | – | – |
| DoD 5220.22 M E C E | O | – | – | – | O | – | O | – | – | – | O | – |
| Bruce Schneier | O | – | O | – | – | – | – | – | – | – | O | – |
| VISITR | O | – | O | O | – | – | O | – | – | – | O | – |
| RCMP TSSIT OPS II | – | – | – | – | – | – | – | – | – | – | O | – |
| Peter Gutmann | O | O | O | O | O | Δ | O | O | O | O | O | – |

**Table 4**
Patterns of typical actions.

| Operation | Pattern |
|---|---|
| File Creation | 0x01→0x04→0x10→0x04→0x01→0x00→0x04→0x20→0x04→(0x04) |
| File Modification | 0x06→0x1f→0x1f→0x04→0x04→0x08 |
| File Renaming | 0x02→0x05→0x01→0x04→0x04→0x04 |
| Simple File Deletion | 0x01→0x04→0x10→0x00→0x04→0x01→0x00→0x06→0x04→0x04→0x04→0x02→0x05→0x01→0x04→ 0x10→0x04→0x01→0x04→0x03→0x04→0x04→0x01→0x04→0x04→0x04→0x04→0x04→0x04→0x08 |
| Permanent File Deletion | 0x0f→0x02→0x02→0x0f→0x02→0x04 |
| Directory Creation | 0x04→0x10→0x01→0x01→0x01→0x0e→0x03→0x04 |
| Directory Renaming | 0x02→0x02→0x01→0x01→0x04→0x04 |
| Simple Directory Deletion | P(Directory Creation)→0x06→0x04→0x04→0x04→0x04→0x03→0x02→0x02→0x01→0x01→0x0e→ 0x04→0x03→0x04→0x04→0x04→0x01→0x04→0x03→0x04→0x04→0x08 |
| Permanent Directory Deletion | 0x02→0x0f →0x02→ 0x0f→0x04→ 0x12 |
| Permanent Directory Deletion Containing Files | 0x0f→0x02→0x02→0x0f→0x02→0x02→0x02→0x0f→0x02→0x0f→0x04→0x12 |

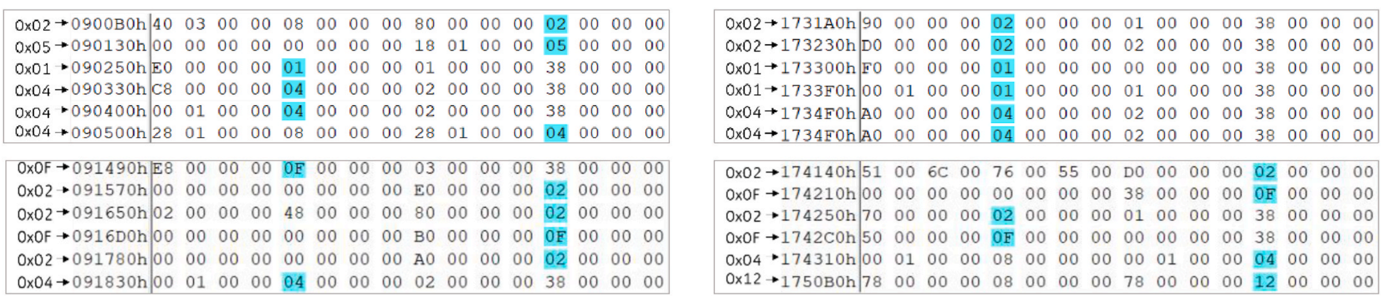



**Fig. 4.** Opcode pattern for File Renaming(Top, Left)/Deletion(Bottom, Left) and Directory Renaming(Top, Right)/Deletion(Bottom, Right).

found before and the new opcodes we identified in version 3.7. Redo record can be used to verify the contents of the task from the metadata in the transaction area under the header using the opcodes, such as filename and time information. An activity is observable in the same or consecutive time value-based last checkpoint. The last checkpoint (a pair of Virtual allocator clocks of the last checkpoint at 0x18 and LSN of the last checkpoint at 0x20) (AlHarbi et al., 2022) was identified in the redo record as shown in Fig. 3. We created a pattern by linking opcodes with the same value.

### 3.2. Fundamental analysis

#### 3.2.1. Analysis of wiping tool

The tools under analysis provide a default delete function for files and directories, supporting the removal of superfluous files, i.e., recycle bin and drive cleanup. We found that the delete function incorporated one or more algorithms, and each tool included at least four different algorithms. In this study, we selected only the algorithms present in at least two of the chosen tools, as illustrated in Table 3.

#### 3.2.2. Analysis of typical deletion pattern

The general deletions have been divided into simple deletion and permanent deletion (see Table 4). Permanent deletion is performed by selecting the file and using the 'Shift + Delete' key. When deleting a directory that contains a file, the file pattern emerges first, followed by the directory pattern.

### 3.3. Analysis of deletion pattern

#### 3.3.1. Analysis of file deletion pattern

Upon analyzing the opcode of the $Logfile after file deletion using the data wiping tools, we identified a common File Modification and File Deletion pattern using the Hex editor. Fig. 4 showcases the File Renaming pattern "0x02→0x05→0x01→0x04→0x04 →0x04" (hereinafter, P(FR)), and the File Deletion pattern "0x0F→0x02→0x02→0x0F→0x02→0x04" (hereinafter, P(FD)). Opcode "0x04″ appears irregularly and is differentiated by being enclosed in parentheses.

#### 3.3.2. Analysis of Directory Deletion pattern

Directory Deletion was executed differently from file patterns. When using a wiping tool to permanently delete a directory, we extracted the common pattern of directory name changes and the pattern of completely deleting a directory using a Hex editor. Fig. 4 displays the Directory Renaming pattern '0x02→0x02→0x01→0x01→0x04→0x04' (hereinafter, P(DR)), and the Directory Deletion pattern '0x02→0x0F→0x02→0x0F→0x04→0x12' (hereinafter, P(DD)).

### 3.4. Wiping tool pattern result

#### 3.4.1. Opcode pattern result

We assigned file and directory names to identify tools and algorithms in the $Logfile. We then extracted all opcodes within the log range specified by user behavior, referring to the final location of the log appearing for each action. Detailed analysis results are tabulated in Table 5. We consolidated and presented the opcode results in patterned form. We employed the '*' symbol for the recursive single opcode and grouped consecutive opcodes using '( )'. If the pattern varies among algorithms, all types are explicitly listed. However, if the pattern remains constant, it is consolidated using the "All" label. Analyzing the opcodes in the $Logfile after deleting a file with a data wiping tool, we identified identical File Changing patterns and File Deletion patterns when the file was permanently deleted with the data wiping tool. We also noticed differences in analysis results for files and directories. In some cases, files were affected by algorithms while directories were not. The iteration count of specific opcodes for Easy File Shredder, Hardwipe, PCShredder, Super File Shredder, and Turbo Shredder was found to be fluctuate depending on the pattern's algorithm. For instance, in Easy File Shredder, the File Renaming Pattern is repeated 36 times in the Gutmann algorithm (overwriting 35 times) and 8 times by overwriting it 7 times. Consequently, the specific number of iterations according to the pattern is the value of 'pass +1'. Easy File Shredder maintains the same iteration count for both File and Directory in the same algorithm. In Super File Shredder, file patterns vary in iteration count depending on

**Table 5**
Data wiping pattern results.

| Algorithm | File Deletion Pattern | Directory Deletion Pattern |
|---|---|---|
| **Easy File Shredder** | | |
| Random, Zero | 0x04*2→P(FR)*4→0x07→0x1f→P(FD) | P(DR)*2→P(DD) |
| GOST P50739 | 0x04*3→P(FR)*4→0x07→0x1f→P(FD) | |
| Air Force 5020, AR380-19, DoD M 3, ITSG 2006, DoD M E | 0x04*4→P(FR)*4→0x07→0x1f→P(FD) | |
| DoD 5220.28 STD, DoD MECE, Bruce Schneier, VISITR | 0x04*8→P(FR)*4→ 0x07→0x1f→P(FD) | P(DR)*8→P(DD) |
| Peter Gutmann | 0x04*36→P(FR)*37→0x07→0x1f→P(FD) | P(DR)*36→P(DD) |
| **File Shredder** | | |
| All | 0x04*2→0x1f→0x04→0x07→0x1f→0x04→P(FR)→P(FD) | P(DR)→0x04→0x03*2→P(DD) |
| **Hardwipe** | | |
| Random, Zero, GOST P50739, DoD M 3, Bruce Schneier, VSITR, Peter Gutmann | ((0x1f)→0x04*(variable))→0x1f→0x04→0x07→0x1f→0x04→P(FR)*3→P(FD) | P(DR)*3→P(DD) |
| **Kernel File Shredder** | | |
| All | 0x06→0x1f*2→0x08*3→P(FR)→0x07→0x1f→P(FD) | P(DR)→P(DD) |
| **PC Shredder** | | |
| DoD M 3 | 0x04*3→P(FR)→0x07→0x1f→P(FD) | P(DR)→0x04→P(DD) |
| DoD MECE | 0x04*7→P(FR)→0x07→0x1f→P(FD) | |
| Peter Gutmann | 0x04*35→P(FR)→0x07→0x1f→P(FD) | |
| **Remo File Eraser** | | |
| All | 0x04→0x1f→0x04*2→P(FR)→0x07→0x1f→P(FD) | P(DR)→P(DD) |
| **Secure Eraser** | | |
| All | 0x06→0x1f→0x08→0x1f→0x04→0x07→0x1f→0x04→P(FR)*9→P(FD) | P(DR)*9→P(DD) |
| **Super File Shredder** | | |
| One, Secure, DoD M | 0x04→0x06→(0x1f*2→0x08→0x04*2→0x06)*2→0x1f*2→0x08→0x04*2 →0x1f→0x04→0x07→0x1f→0x04→P(FR)→P(FD) | P(DR)→P(DD) |
| Peter Gutmann | 0x04→0x06→(0x1f*2→0x08→0x04*2→0x06)*14→0x1f*2→0x08→ 0x04*2→0x1f→0x04→0x07→0x1f→0x04→P(FR)→P(FD) | |
| **TurboShredder** | | |
| Zero, One, Secure Random | 0x04*(variable)→P(FR)*pass→0x04*(pass+1)→0x07→0x1f→P(FD) | Not Available |
| Peter Gutmann | 0x04*(variable)→P(FR)→0x04*37→0x07→0x1f →P(FD) | |
| **WipeFile** | | |
| All | 0x04*2→0x07→0x1f→0x04→P(FR)→0x04*3→P(FD) | 0x04→0x03*2→P(DR)→0x04→ 0x04→(0x03*2→0x04*2)*3→P (DD) |
| **xShredder** | | |
| All | 0x06→0x04*4→(0x04)→P(FR)→0x04(0x04)*6→0x07→0x04→P(FD) | Not Available |
| **XT File Shredder** | | |
| All | 0x04→0x1f→0x04→0x07→0x1f→0x04→P(FR)→P(FD) | P(DD) |

**Table 6**
File/directory name transition according to renaming pattern.

| Tool | Filename pattern + (extension pattern) | Directory |
|---|---|---|
| Easy File Shredder | A random array of alphabets, symbols and number | A random array of alphabets and number |
| File Shredder | Random number + (. repeat 'Z') | Random number |
| Hardwipe | Random lowercase alphabets | Random lowercase alphabets |
| Kernel File Shredder | Repeat one random uppercase letter | Repeat one random uppercase letter |
| PC Shredder | temp + Random 11 number | temp + Random 11 number |
| Remo File Eraser | Repeat 'x' + (. Repeat 'x') | Repeat 'x' |
| Secure Eraser | A random array of alphabets or number | A random array of alphabets or number |
| Super File Shredder | 1070E08F + Random number | 0. + four-digit random number |
| Turbo shredder | A random array of alphabets or number | – |
| Wipe File | Random alphabets and number + (. random letter) | Random alphabets and number |
| xShredder | Repeat one random lowercase letter | – |
| XT File Shredder | Random lowercase alphabets + (.random alphabets) | – |

the algorithm, but in directory patterns, all patterns remain consistent regardless of the algorithm. Hardwipe has different patterns depending on the file size. When we create a file size of 10 KB in Hardwipe, the pattern is as follows: '0x04→(0x1f→0x04)*69 → 0x1f→0x04→0x07→0x1f→0x04→P(FR) *3→P(FD)'. However, when the file size is set to 50 KB, the pattern becomes '[4(N * More than 20)→0x1f→0x04→0x07→0x1f→0x04→P(FR)*3→P(FD)]'. In this case, it is confirmed that N is variable depending on the file size.

**Table 7**
Summary of pattern result.

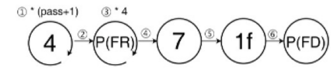
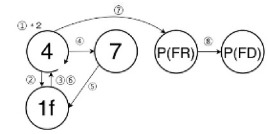
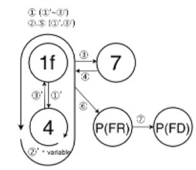
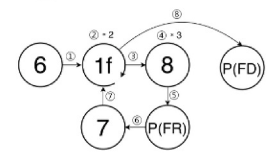
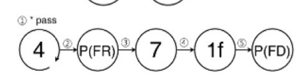
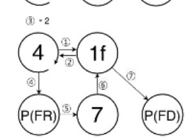
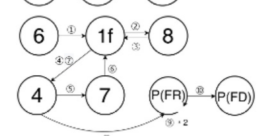
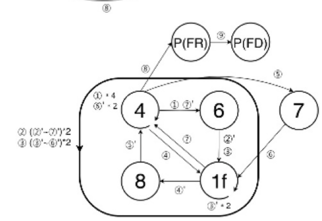
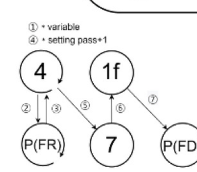
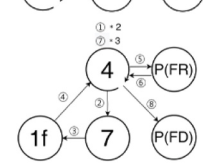
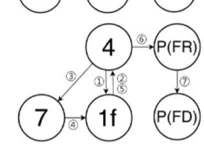
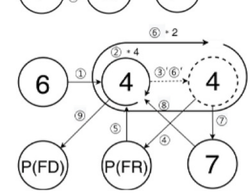
| Tool | Node, Edge | Diagram |
|---|---|---|
| Easy File Shredder | $n_p(EasyFileShredder) = \{0x04, 0x07, 0x1f, P(FR), P(FD)\}$ <br> $e_p(EasyFileShredder) = \{0x04 \rightarrow P(FR), P(FR) \rightarrow 0x07, 0x07 \rightarrow 0x1f, 0x04 \rightarrow 0x1f, 0x1f \rightarrow P(FD)\}$ |  |
| File Shredder | $n_p(FileShredder) = \{0x04, 0x07, 0x1f, P(FR), P(FD)\}$ <br> $e_p(FileShredder) = \{0x04 \rightarrow 0x07, 0x07 \rightarrow 0x1f, 0x1f \rightarrow 0x04, 0x04 \rightarrow 0x1f, 0x04 \rightarrow P(FR), P(FR) \rightarrow P(FD)\}$ |  |
| Hardwipe | $n_p(Hardwipe) = \{0x04, 0x07, 0x1f, P(FR), P(FD)\}$ <br> $e_p\ (Hardwipe) = \{0x1f \rightarrow 0x04, 0x04 \rightarrow 0x07, 0x07 \rightarrow 0x1f, 0x1f \rightarrow P(FR), 0x04 \rightarrow P(FR), P(FR) \rightarrow P(FD)\}$ |  |
| Kernel File Shredder | $n_p(KernelFileShredder) = \{\ 0x06, 0x07, 0x08, 0x1f, P(FR), P(FD)\}$ <br> $e_p(KernelFileShredder) = \{0x06 \rightarrow 0x1f, 0x1f \rightarrow 0x08, 0x07 \rightarrow 0x1f, 0x1f \rightarrow P(FR), 0x08 \rightarrow P(FR), 0x1f \rightarrow P(FD)\}$ |  |
| PC Shredder | $n_p(PCShredder) = \{0x04, 0x07, 0x1f, P(FR), P(FD)\}$ <br> $e_p(PCShredder) = \{0x04 \rightarrow P(FR), P(FR) \rightarrow 0x07, 0x07 \rightarrow 0x1f, 0x1f \rightarrow P(FD)\}$ |  |
| Remo File Eraser | $n_p(RemoFileEraser) = \{0x04, 0x07, 0x1f, P(FR), P(FD)\}$ <br> $e_p(RemoFileEraser) = \{\ 0x04 \rightarrow 0x1f, 0x1f \rightarrow 0x04, 0x07 \rightarrow 0x1f, 0x04 \rightarrow P(FR), P(FR) \rightarrow 0x07, 0x1f \rightarrow P(FD)\}$ |  |
| Secure Eraser | $n_p(Secure\ Eraser) = \{0x04, 0x06, 0x07, 0x08, 0x1f, P(FR), P(FD)\}$ <br> $e_p\ (SecureEraser) = \{0x06 \rightarrow 0x1f,\ 0x1f \rightarrow 0x04,\ 0x07 \rightarrow 0x1f,\ 0x1f \rightarrow 0x04,\ 0x04 \rightarrow 0x07,\ 0x07 \rightarrow 0x1f,\ 0x04 \rightarrow P(FR), P(FR) \rightarrow P(FD)\}$ |  |
| Super File Shredder | $n_p(SuperFileShredder) = \{0x04, 0x06, 0x07, 0x08, 0x1f, P(FR), P(FD)\}$ <br> $e_p(SuperFileShredder) = \{0x04 \rightarrow 0x06, 0x06 \rightarrow 0x1f, 0x1f \rightarrow 0x08, 0x08 \rightarrow 0x04, 0x1f \rightarrow 0x04, 0x04 \rightarrow 0x1f, 0x07 \rightarrow 0x1f, 0x04 \rightarrow 0x07, 0x04 \rightarrow P(FR), P(FR) \rightarrow P(FD)\}$ |  |
| Turbo Shredder | $n_p(TurboShredder) = \{0x04, 0x07, 0x1f, P(FR), P(FD)\}$ <br> $e_p(TurboShredder) = \{0x04 \rightarrow P(FR), P(FR) \rightarrow 0x04, 0x04 \rightarrow 0x07, 0x07 \rightarrow 0x1f, 0x1f \rightarrow P(FD)\}$ |  |
| WipeFile | $n_p(WipeFile) = \{0x04, 0x07, 0x1f, P(FR), P(FD)\}$ <br> $e_p(WipeFile) = \{0x04 \rightarrow 0x07, 0x07 \rightarrow 0x1f, 0x1f \rightarrow 0x04, 0x04 \rightarrow P(FR), P(FR) \rightarrow 0x04, 0x04 \rightarrow P(FD)\}$ |  |
| XT File Shredder | $n_p(XTFileShredder) = \{0x04, 0x07, 0x1f, P(FR), P(FD)\}$ <br> $e_p(XTFileShredder) = \{0x04 \rightarrow 0x1f,\ 0x1f \rightarrow 0x04,\ 0x04 \rightarrow 0x07,\ 0x07 \rightarrow 0x1f,\ 0x04 \rightarrow P(FR),\ P(FR) \rightarrow P(FD)\}$ |  |
| xShredder | $n_p(xShredder) = \{0x04, (0x04), 0x06, 0x07, P(FR), P(FD)\}$ <br> $e_p(xShredder) = \{0x06 \rightarrow 0x04, 0x04 \rightarrow P(FR), (0x04) \rightarrow P(FR), 0x04 \rightarrow (0x04), P(FR) \rightarrow 0x04, 0x04 \rightarrow P(FR), 0x04 \rightarrow 0x07, (0x04) \rightarrow 0x07, 0x04 \rightarrow P(FD)\}$ |  |

**Table 8**
Design of verification.

| No | Validation method | File type | Number of files | Size of file (KB) | Repetition |
|---|---|---|---|---|---|
| 1 | Fixed-size validation | Create Dummy File using script | 70 | 10 | 2 |
| 2 | Fixed-size validation | Create Dummy File using script | 70 | 50 | 2 |
| 3 | Fixed-size validation | Create Dummy File using script | 70 | 100 | 2 |
| 4 | Fixed-size validation | Create Dummy File using script | 70 | 10,240 | 1 |
| 5 | Variable-size validation | PDF, XLS, DOC files with variable size | 70 | 8~36,235 | 3 |
| Total number of files | | | | | 700 |
| Total number of Logfiles | | | | | 10 |

### *3.4.2. Filename pattern result*

Transitions in File/Directory Names can also be observed in the metadata area where the Renaming pattern appears. As illustrated in Table 6, these name transitions exhibited distinct characteristics for each tool and were unaffected by the algorithms employed. In some instances, such as with Kernel File Shredder, these transitions were recurrently composed of capital letters. Conversely, others, like Hardwipe, comprised random alphabets. Identifying the variations in file name changes collectively enables the differentiation of opcode patterns even in instances of duplication.

### *3.5. Data wiping tool patterns*

As shown in Table 7, the experimental results are represented through a diagram with nodes and edges. Opcodes within each sequence were set as nodes, and the edges connecting the nodes were used to denote the order and repetition count. This diagram comprehensively represents the existing pattern by tracking overall opcode changes and identifying specific alterations. When a node repeats, it's represented in a rotational form. The connecting link (edge) and (node) for '0x04' are shown as a dotted line to distinguish it from 0x04. Interestingly, while most tools had '0x04', '0x1f', and '0x07', Kernel File Shredder, Super File Shredder, and xShredder only include '0x06' and '0x08'. They all had different diagrams, which implies that each tool exhibits unique characteristics.

## 4. Implementation and verification

### *4.1. Implementation*

To assess our approach, we developed a tool using the proposed method of opcode analysis based on the $Logfile structure and detection of data wiping tools. We utilized a Python 3.10 development environment with Pyside and PyQt5 GUI (URL: https://github.com/jamemanionda/ReFS_Detector). The tool's operation is as follows: 1) Upon uploading the $Logfile, 2) the log area is analyzed, yielding a result comprised of the file, time, and opcodes appearing in time units. The resulting fields are structured as [Detect Tool Name] – [Filename] – [Date] – [Pattern]. The Filename and Date fields were extracted from the time and filenames of the Data area's metadata. The Detect Tool Name field stores specific tool patterns, and after extracting the opcodes of the input file, 3) compares these with the analyzed patterns of the 12 tools, 4) indicating the tool name if a match is found. Moreover, 5) if a specific tool is not utilized, but the Renaming (P(FR)/P(DR)) or Deletion (P(FD)/P(DD)) pattern is embedded, the program raises an alert for potential data wiping tool usage. This program can also be updated to include patterns for new tools beyond the 12 deletion tools already incorporated.

### *4.2. Verification*

To gauge the accuracy of the implemented program in detecting the behavior of data wiping tools, we conducted a cross-validation process using the patterns described in Section 4.1. We used an open-source script (Matuzalem, 2022) to generate dummy files of sizes 10 KB, 50 KB, 100 KB, and 10,240 KB, in order to examine the potential impact of file size changes, similar to Hardwipe. Additionally, to assess the effectiveness of the detection capability, we utilized a dataset of PDF, XLS, and DOC files obtained from Digital Corpora (Simson Garfinkel, 2020) (Digital Corpora, 2021). By randomly applying files of different sizes from this dataset, we could verify the detection capability. For each repetition, we created a total of 70 files to be deleted one by one using each tool and algorithm combination. This allowed us to perform verification using the number of files as shown in Table 8. The dummy file was directly created on the ReFS partition through the program, and for the pdf/xls/doc files of various sizes, only deletion was performed on the already created files. All the logs were processed for deletion in the order specified in Appendix A.

As a result, we were able to successfully detect all the tools consistently. The generated log files, along with the respective tools, can be found in ReFS_Detector repository. However, in the case of Turbo's Gutmann, the log file size was exceptionally large, so we added it separately as it overwrote the existing $Logfile.

## 5. Conclusion

Permanent deletion primarily aims not only to destroy evidence but also to protect privacy. Accordingly, numerous studies have been conducted on permanent deletion in various file systems. However, only a few have specifically focused on ReFS. While previous research has addressed file creation, modification, and deletion in ReFS, it only considered standard deletion. In contrast, our work has honed in on the anti-forensic issue, concentrating on the deletion behavior within wiping tools.

To detect user behavior concerning data wiping tools in ReFS, we focused that actions can be identified from the $Logfile. It involves using the $Logfile to detect behavior for 12 wiping tools and identifying each tool. We found a variety of patterns, yielding different results for each algorithm, and tools with the same pattern exist regardless of the algorithm. As for the P(File Deletion), both the deletion pattern and the renaming pattern were common across all tools, nearly consistent with the file renaming and file deletion patterns addressed in previous research. Therefore, we discovered that file renaming and file deletion patterns always remain when files are deleted using complete deletion tools. Our findings also revealed that the resulting data can be influenced by wiping tools, which employ deletion algorithms to overwrite files multiple times based on specific algorithms. We also uncovered the name transition in the metadata area, recorded along with the name change pattern, according to the tool used. Finally, we implemented a tool capable of detecting and identifying a data wiping tool using the analysis method and the extracted pattern. The wiping tool used was appropriately detected when we checked through the program we developed after an actual wipe. Future releases of ReFS may impact our findings. Therefore, regular updates to the tool will be necessary, as there might be minor differences in results based on the version of ReFS as we see the difference between 3.4 and 3.7.

Deletion patterns are not merely an analytical result but can also be used effectively in investigations. By storing each tool's wiping pattern in a database, the function of identifying the data wiping tool can prove wiping action by checking the results of the opcode automatic analysis. The ReFS $Logfile size can vary based on the file system size and operation frequency, and it can be adjusted through settings. If the $Logfile space becomes full, older behavior may be challenging to analyze as the log file will be reused. The tool that we implemented will help digital forensic examiners determine whether the data wiping tool was used. We recommend correlating the $Logfile findings together with other artifacts, not relying on a single source. Detecting wiping patterns during an investigation could indicate potential evidence deletion. Both the analysis method and results against the data wiping tool proposed in this study are anticipated to aid in uncovering and identifying permanent delete behavior in dedicated digital forensic investigation.

## Appendix

**Table A.9**
Data used for variable size validation

| Filename | Extention | Size (KB) |
|---|---|---|
| 01_01_Easy_Random(70) | doc | 40 |
| 01_02_Easy_Zero(69) | xls | 7699 |
| 01_03_Easy_GOST(68) | pdf | 144 |
| 01_04_Easy_Airforce(67) | doc | 57 |
| 01_05_Easy_AR380(66) | xls | 2319 |
| 01_06_Easy_DoDM3(65) | pdf | 9051 |
| 01_07_Easy_DoDME(64) | doc | 55 |
| 01_08_Easy_ITSG2006(63) | xls | 27929 |
| 01_10_Easy_DoD28STD(62) | doc | 172 |
| 01_11_Easy_DoD22MECE(61) | xls | 5450 |
| 01_12_Easy_Bruce(60) | pdf | 222 |
| 01_13_Easy_VISITR(59) | doc | 177 |
| 01_15_Easy_Gutmann(58) | pdf | 9731 |
| 02_01_FileShred_One(57) | doc | 66 |
| 02_02_FileShred_Secure(56) | xls | 34 |
| 02_03_DoDM3(55) | pdf | 2213 |
| 02_03_Gutmann(54) | doc | 175 |
| 03_01_Hard_Random(53) | xls | 80 |
| 03_02_Hard_Zero(52) | pdf | 222 |
| 03_03_Hard_GOST(51) | doc | 57 |
| 03_04_Hard_DoDM3(50) | xls | 3707 |
| 03_05_Hard_Bruce(49) | pdf | 1047 |
| 03_06_Hard_VISITR(48) | doc | 105 |
| 03_07_Hard_Gutmann(47) | xls | 70 |
| 04_01_Kernel_Zero(46) | pdf | 160 |
| 04_02_Kernel_HMG(45) | doc | 173 |
| 04_03_Kernel_GOST(44) | xls | 51 |
| 04_04_Kernel_AirForce(43) | pdf | 593 |
| 04_05_Kernel_AR380(42) | doc | 13838 |
| 04_06_Kernel_DoDM3(41) | xls | 99 |
| 04_07_Kernel_VISITR(40) | pdf | 8 |
| 04_08_Kernel_Gutmann(39) | doc | 20 |
| 05_01_PC_DoDM3(38) | xls | 90 |
| 05_02_PC_DoDMECE(37) | pdf | 64 |
| 05_03_PC_Gutamann(36) | doc | 1299 |
| 06_01_Remo_Random(35) | xls | 14303 |
| 06_02_Remo_Zero(34) | pdf | 43 |
| 06_03_Remo_DoDM3(33) | doc | 138 |
| 07_01_Secure_Random(32) | xls | 2757 |
| 07_02_Secure_DoDME(31) | pdf | 268 |
| 07_03_Secure_DoD22MECE(30) | doc | 112 |
| 07_04_Secure_VISITR(29) | xls | 73 |
| 07_05_Secure_Gutmann(28) | pdf | 927 |
| 08_01_Super_One(27) | doc | 30 |
| 08_02_Super_Secure(26) | xls | 2679 |
| 08_03_Super_DoDM3(25) | pdf | 1216 |
| 08_04_Super_Gutmann(24) | doc | 136 |
| 09_01_Wipe_One(23) | doc | 28 |
| 09_02_Wipe_Secure(22) | xls | 9252 |
| 09_03_Wipe_Random(21) | pdf | 36235 |
| 09_04_Wipe_Zero(20) | doc | 545 |
| 09_05_Wipe_Gutmann(19) | xls | 3682 |
| 10_01_xShredder_HMG(18) | pdf | 383 |
| 10_02_xShredder_GOST(17) | doc | 32 |
| 10_03_xShredder_AirForce(16) | xls | 7699 |
| 10_04_xShredder_AR380(15) | pdf | 591 |
| 10_05_xShredder_DoDM3(14) | doc | 103 |
| 10_06_xShredder_DoDMECE(13) | xls | 2319 |
| 10_07_xShredder_Bruce(12) | pdf | 4916 |
| 10_08_xShredder_VISITR(11) | doc | 90 |

(*continued on next page*)

**Table A.9** (*continued*)

| Filename | Extention | Size (KB) |
|---|---|---|
| 10_09_xShredder_RCMP(10) | xls | 27929 |
| 10_10_xShredder_Gutmann(09) | pdf | 177 |
| 11_01_XT_DoDM3(08) | doc | 425 |
| 11_02_XT_DoDME(07) | xls | 5450 |
| 11_03_XT_One(06) | doc | 95 |
| 12_01_Turbo_One(05) | xls | 2790 |
| 12_02_Turbo_Secure(04) | pdf | 158 |
| 12_03_Turbo_Random(03) | doc | 6411 |
| 12_04_Turbo_Zero(02) | xls | 3065 |
| 12_05_Turbo_Gutmann(01) | pdf | 46 |

**Table A.10**
Wiping Tool List and Functions(O: Available, -: Unavailable, Δ: Delete only subdir)

| Function/Tool | Easy File Shredder | File Shredder | Hard Wipe | Kernel File Shredder | PC Shredder | Remo File Eraser | Secure Eraser | Super File Shredder | Turbo Shredder | Wipe File | xShredder | XT File Shredder |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Wipe File | O | O | O | O | O | O | O | O | O | O | O | O |
| Wipe Directory | O | O | O | O | O | O | O | O | – | O | Δ | O |
| Repeat Option | O | – | – | – | – | – | – | – | – | – | – | – |
| Speed Option | – | – | O | – | – | – | – | – | – | – | – | – |
| Zero After Wipe | O | – | | – | – | – | – | O | O | – | – | – |
| Cleanup Drive/Disk | O | – | O | – | – | O | – | – | – | – | – | – |
| Cleanup Recycle bin | – | – | O | O | – | O | – | – | – | – | O | O |
| Cleanup Pagefile | – | – | O | O | – | – | – | – | – | – | O | – |
| Cleanup Free Space | O | O | O | – | – | O | – | – | – | – | O | – |
| Cleanup System File | – | – | – | O | – | – | O | – | – | – | – | – |
| Cleanup Registry | – | – | – | – | – | – | O | – | – | – | – | – |
| Cleanup Useless File | – | – | – | O | – | – | – | – | – | – | – | – |
| MultiTask | O | O | – | O | O | O | O | O | O | O | O | O |
| Schedule Task | – | – | – | O | – | O | O | – | – | – | O | – |
| Log | – | – | O | O | – | – | O | – | – | O | – | O |
| Report | – | – | – | – | – | – | O | – | – | – | – | – |

**Table A.11**
Data Wiping Pattern Results (ReFS version 3.4)

| Algorithm | File Deletion Pattern | Directory Deletion Pattern |
|---|---|---|
| Easy File Shredder | | |
| Random, Zero | P(FR)*3→P(FD) | P(DR)*2→P(DD) |
| GOST P50739-95 | P(FR)*4→P(FD) | |
| Air Force 5020, AR380-19, DoD 5220.M 3, ITSG2006 | | |
| DoD 5220.M E | P(FR)*5→P(FD) | P(DR)*4→P(DD) |
| DoD 5220.28 STD, DoD 5220.22 M E C E, Bruce Schineier, German VISITR | P(FR)*9→P(FD) | P(DR)*8→P(DD) |
| Peter Gutmann | P(FR)*37→P(FD) | P(DR)*36→P(DD) |
| File Shredder | | |
| All | 0x04*3→P(FR)→P(FD) | P(DR)→0x04→0x03*2→P(DD) |
| Hardwipe | | |
| Zero | 0x06→0x04*2 → 0x08→0x04*3 → 0x07→0x04*2→(0x04)→ (FR)*3→P(FD) | P(DR)*3→P(DD) |
| DOD 5220.22 M | 0x06→0x04*2 → 0x08→0x04*5 → 0x07→0x04*2→(0x04)→P(FR)*3→P(FD) | |
| GOST P50739 | 0x06→0x04*2 → 0x08→0x04*7 → 0x07→0x04*2→(0x04)→P(FR)*3→P(FD) | |
| Bruce Schneier, VSITR | 0x06→0x04*2 → 0x08→0x04*15 → 0x07→0x04*2 (0x04)→P(FR)*3→P(FD) | |
| Peter Gutmann | 0x06→0x04*2 → 0x08→0x04*71 → 0x07→0x04*2→(0x04)→P(FR)*3→P(FD) | |
| Kernel File Shredder | | |
| All | 0x04→0x04→P(FR)→P(FD) | P(DR)→P(DD) |
| PC Shredder | | |
| All | P(FR)→P(FD) | P(DR)→0x04→P(DD) |

(*continued on next page*)

**Table A.11** (*continued*)

| Algorithm | File Deletion Pattern | Directory Deletion Pattern |
|---|---|---|
| Remo File Eraser | | |
| All | 0x04→P(FR)→P(FD) | P(DR)→P(DD) |
| Secure Eraser | | |
| All | 0x04→P(FR)*9→P(FD) | P(DR)*9→P(DD) |
| Super File Shredder | | |
| One, DoD 5220 22M, Secure Eraser | 0x04→{0x06→0x04*2 → 0x08→0x04*2}(3)→0x04*2 → 0x07→0x04*2→(0x04)→P(FR)→P(FD) | P(DR)→P(DD) |
| Gutmann | 0x04→{0x06→0x04*2 → 0x08→0x04*2}(15)→0x04*2 → 0x07→0x04*2→(0x04)→P(FR)→P(FD) | |
| TurboShredder | | |
| Zero, One, Secure | 0x04→P(FR)*4 → 0x04(0x04)*2→(0x04) →P(FD) | Not Available |
| Peter Gutmann | 0x04→P(FR) →(0x04) →P(FD) | |
| WipeFile | | |
| All | 0x04*2→P(FR)→0x04*3→P(FD) | 0x04→0x03*2→P(DR)→0x04→0x04→[0x03*2 → 0x04*2]*3→P(DD) |
| xShredder | | |
| All | 0x06→0x04*4→(0x04)→P(FR)→0x04(0x04)*6 → 0x07→0x04→P (FD) | Not Available |
| XT File Shredder | | |
| All | 0x04*3→P(FR)→P(FD) | P(DD) |

## References


AlHarbi, R., AlZahrani, A., Bhat, W.A., 2022. Forensic analysis of anti-forensic file wiping tools on Windows. J. Forensic Sci. 67, 562. https://doi.org/10.1111/1556-4029.14907.

Bitraser. Supports Global Algorithms for Media Sanitization. URL. https://www.bitraser.com/data-erasure-standards.php.

Brinkmann, Martin, 2023. Windows 11 is Getting Refs Support. URL: http://www.ghacks.net/2023/01/24/windows-11-is-getting-refs-support. xShredder. URL https://sourceforge.net/projects/xshredder/.

Bundesamts fr Sicherheit in der Informationstechnik, 2004. Richtlinien zum Geheimschutz von Verschlusssachen beim Einsatz von Informationstechnik.

Cho, G.S., 2018. A Steganographic Data Hiding Method in Timestamps by Bit Correction, vol. 23, p. 75. https://doi.org/10.9708/jksci.2018.23.08.075.

Digital Corpora. URL. https://downloads.digitalcorpora.org/corpora/files/CC-MAIN-2021-31-PDF-UNTRUNCATED/zipfiles/0000-0999/.

Easy file shredder. URL. https://www.easyfileshredder.com/features.php.

FileShredder. URL: https://www.fileshredder.org/.

Garfinkel, Simson, 2020. Digital Corpora. URL: https://downloads.digitalcorpora.org/corpora/files/govdocs1/by_type/.

Gosstandart of Russia, 1995. GOST R 50739-50795.

Gudadhe, S., Deoghare, R., Dhade, P., 2015. Window ReFS file system: a study. Int. J. Adv. Res. Comput. Commun. Eng. 4 https://doi.org/10.17148/IJARCCE.2015.44138.

Gutmann, P., 1996. Secure deletion of data from magnetic and solid-state memory. In: Sixth USENIX Security Symposium, p. 77.

Hardwipe. https://hardwipe.en.softonic.com/.

Horsman, G., 2021. Digital tool marks (DTMs): a forensic analysis of file wiping software. Aust. J. Forensic Sci. 53, 96. https://doi.org/10.1080/00450618.2019.1640793.

Jason Evans, 2022. Convicted Sex Offender used Data. URL: http://www.walesonline.co.uk/news/wales-news/convicted-sex-offender-used-data-24874840.

Jones, A., Afrifa, I., 2020. An evaluation of data erasing tools. J. Digit. Forensic Secur. Law 15. https://doi.org/10.15394/jdfsl.2020.1615.

Kakasoft. Super file shredder. URL: https://super-file-shredder.en.softonic.com./.

Kernel file shredder. URL: https://www.nucleustechnologies.com/file-shredder.html.

Kim, S.H., 2019. Analysis of ReFS Data Storage Principles and Study of Deleted Data Recovery Technology Using Metadata". Korea University.

Kim, M.H., Lee, S.J., 2015. Method of estimating the deleted time of applications using Amcache.hve. J. Kor. Inst. Inf. Secur. Cryptol. 25, 573.

Kim, D.G., Park, S.H., Jo, O., 2020. Analyzing past user history through recovering deleted $UsnJrnl file. J. Converg. Inf. Technol. 10, 23.

Lee, J.S., 2020. The criminal meaning of the evidence destruction and that of the criminal case against another. Kor. Lawyers Assoc. J. 69, 192. https://doi.org/10.17007/klaj.2020.69.1.007.

Lee, S.H., Choi, H.Y., Park, J.H., Lee, S.J., 2019. Analysis of ReFS journaling file. J. Digit. Forensic 13, 189. https://doi.org/10.22798/kdfs.2019.13.3.189.

Lee, S.H., Park, J.H., Hwang, H.U., Lee, S.Y., 2021. Forensic analysis of ReFS journaling. Forensic Sci. Int.: Digit. Invest. 38, 301136 https://doi.org/10.1016/j.fsidi.2021.301136.

Maneas, S., Mahdaviani, K., Emami, T., Schroeder, B., 2021. Reliability of SSDs in Enterprise Storage Systems: A Large-Scale Field Study, vol. 17, p. 27.

Matuzalem (Mat) Muller dos Santos, 2022. URL: https://github.com/matuzalemmuller/dummy-files-creator.

Microsoft. URL. https://learn.microsoft.com/en-us/windows-server/storage/refs/refs-overview.

Nordvik, R., Georges, H., Toolan, F., Axelsson, S., 2019. Reverse engineering of ReFS. Digit. Invest. 30, 127. https://doi.org/10.1016/j.diin.2019.07.004.

PCShredder, 2008. URL: http://www.pcshredder.com/.

Prade, P., Grob, T., Dewald, A., 2019. Forensic Analysis of the Resilient File System (ReFS) Version 3.4. Friedrich-Alexander-Universitat Erlangen-Nurnberg, Dept. of Computer Science. https://d-nb.info/1201551625/34.

Prade, P., Grob, T., Dewald, A., 2020. Forensic analysis of the Resilient File system (ReFS) version 3.4. Forensic Sci. Int.: Digit. Invest. 32, 300915 https://doi.org/10.1016/j.fsidi.2020.300915.

RCMP G2-003, 2003. Hard Drive Secure Information Removal and Destruction Guidelines.

Remo software. URL: https://www.remosoftware.com/remo-file-eraser.

Russinovich, M., 2012. Windows Internals, Part 2.

Savoldi, A., Piccinelli, M., Gubian, P., 2012. A statistical method for detecting on disk wiped areas. Digit. Invest. 8, 194. https://doi.org/10.1016/j.diin.2011.06.005.

Schneier, B., 1995. Applied cryptography. In: Protocols, Algorithms, and Source Code in C, second ed. John Wiley & Sons, USA.

Secure eraser. URL: https://secure-eraser.en.softonic.com./.

Shin, Y.H., Cheon, J.Y., Kim, J.S., 2016. Study on recovery techniques for the deleted or damaged event log(EVTX) files. J. Kor. Inst. Inf. Secur. Cryptol. 26, 387. https://doi.org/10.13089/JKIISC.2016.26.2.387.

Smith, C., Dietrich, G., Kwang, K., Raymon, C., 2017. Identification of forensic artifacts in VMWare virtualized computing. Secur. Priv. Commun. Netw. 85 https://doi.org/10.1007/978-3-319-78816-6_7.

Turbo shredder. URL. https://sourceforge.net/projects/turboshredder/.

U.S. Air Force, 1998. System Security Instruction.

U.S. Army, 1998. Army Regulation 380-19.

U.S. Defense Security Services, 2007. Clearing and Sanitization Matrix.

U.S. Dept. of energy, 2005. DOE M 205.1-2, Clearing, Sanitization, and Destruction of Information System Storage Media, Memory Devices, and Related Hardware Manual.

U.S. Dept. of the Navy. NAVSO 5239-26, Remanence.

Woolwine, Jason, 2022. Virginia Beach Man. URL: http://www.wtkr.com/news/virginia-beach-man-sentenced-27-years-for-creating-video-of-himself-sexually-abusing4-year-old-boy.

Washington, D.C., 1978. DoD Instruction 5220.28, Application of Special Eligibility and Clearance Requirements in the SIOP-ESI Program for Contractor Employees.

Wipe file. URL: https://www.gaijin.at/en/software/wipefile.

xShredder, URL : sourceforge.net/p/xshredder/wiki/Home.

XT file shredder. URL: https://www.lizard-labs.com/.