Volume 11, Issue 4, April – 2026 International Journal of Innovative Science and Research Technology
ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr738

# MetaRecoverX: Recovery of Deleted Data and Associate Metadata from XFS and Btrfs Filesystems

Jagendra Singh Chaudhary[1]; Lucky Panchal[1]; Mayank Tak[1]; Milan Kumar[1]

[1]Department of Information Technology, Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur-302017, India

Publication Date: 2026/04/18

**Abstract: The unintentional loss of digital data due to acciden-tal file deletion, filesystem corruption, or storage device failure represents a critical challenge in the domain of digital forensics and data recovery. While conventional recovery approaches tend to address either file data or metadata in isolation, the simultaneous and accurate restoration of both remains largely underexplored, particularly for modern Linux filesystems such as XFS and Btrfs. This paper presents MetaRecoverX, a purpose-built, Python-based forensic tool that performs deep file carving across more than sixteen file types using magic byte signatures, coupled with structured metadata extraction including timestamps, file permissions, SHA-256 cryptographic hash verification, EXIF data, and document-embedded attributes. The system integrates a command-line interface alongside a PyQt6-based graphical user interface, enabling forensic professionals and general users alike to conduct thorough recovery sessions. Automated generation of detailed PDF and CSV investigation reports further distinguishes MetaRecoverX from existing recovery frameworks. Experimental evaluation conducted on an Ubuntu 24.04 LTS environment demonstrates recovery rates exceeding ninety-two percent for XFS partitions and approximately eighty-five percent for Btrfs partitions, with consistent metadata restoration across diverse file categories. MetaRecoverX contributes a unified, extensible, and practically deployable solution to the evolving landscape of filesystem forensics.**

***Keywords:*** *Digital Forensics, File Carving, Metadata Recov-ery, XFS Filesystem, Btrfs Filesystem, Python, Data Recovery, SHA-256 Integrity Verification, PyQt6, Forensic Reporting.*

**How to Cite:** Jagendra Singh Chaudhary; Lucky Panchal; Mayank Tak; Milan Kumar (2026) MetaRecoverX: Recovery of Deleted Data and Associate Metadata from XFS and Btrfs Filesystems. *International Journal of Innovative Science and Research Technology*, 11(4), 997-1003. https://doi.org/10.38124/ijisrt/26apr738

## I. INTRODUCTION

The integrity and recoverability of digital data have become foundational concerns across both personal and professional computing environments. As organizations and individuals increasingly rely on Linux-based storage systems, the risk of data loss through accidental deletion, filesystem corruption, or deliberate file removal has grown substantially. Unlike Windows-oriented filesystems where recovery tools are com-paratively mature and widely available, the forensic recovery of files from advanced Linux filesystems such as XFS and Btrfs presents considerably greater technical complexity.

XFS is a high-performance journaling filesystem originally developed by Silicon Graphics and now widely adopted in enterprise Linux distributions due to its scalability and robustness. Btrfs, by contrast, represents a more modern approach to filesystem design, incorporating copy-on-write semantics, built-in snapshot support, and dynamic volume management. While these architectural features offer significant advantages in terms of data reliability and operational flexibility, they simultaneously complicate the process of recovering deleted files and their associated metadata. In particular, the copy-on-write mechanism in Btrfs and the journal-based structure of XFS introduce unique challenges when attempting to reconstruct deleted file records without overwriting residual data.

Existing recovery solutions, including tools such as xfs_repair, btrfs restore, and testdisk, primarily address filesys-tem repair and data block retrieval. They do not offer compre-hensive metadata recovery, automated forensic reporting, or sup-port for carving multiple file types by signature. Furthermore, none of these tools provide an integrated graphical interface that makes forensic analysis accessible to non-specialist users. This paper introduces MetaRecoverX, a forensic data recov-ery framework developed entirely in Python that addresses these limitations through a unified architecture combining file carving, metadata extraction, cryptographic integrity verification, and automated report generation. The tool supports more than

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr738

sixteen file types through magic byte identification, extracts rich metadata including EXIF attributes from image files and embedded properties from PDF and Office documents, and produces structured investigation reports in both PDF and CSV formats. A PyQt6-based graphical user interface accompanies the command-line interface, ensuring usability across different user profiles. MetaRecoverX was developed and validated on Ubuntu 24.04 LTS operating on a dual-boot system with an approximately eighty-five gigabyte dedicated Linux partition. The remainder of this paper is structured as follows. Section

II reviews related work in data and metadata recovery for Linux filesystems. Section III describes the problem context and technical challenges associated with XFS and Btrfs recovery. Section IV presents the proposed MetaRecoverX framework and its architectural components. Section V outlines the experimental setup and evaluation methodology. Section VI reports and analyses the results. Section VII concludes the paper and identifies directions for future work.

## II. RELATED WORK / LITERATURE REVIEW

Research into filesystem forensics and deleted file recovery has evolved considerably over the past two decades, driven by growing demand in both law enforcement digital investigation and enterprise data management. This section surveys relevant prior work across three thematic areas that directly inform the design and contribution of MetaRecoverX.

➢ *XFS Filesystem Forensics and Recovery*

XFS has received growing attention in the forensic research community due to its widespread deployment in enterprise and server environments. Wang [1] examined data storage structures within XFS and proposed methods for recovering deleted file records by analysing inode allocation patterns and journal transactions. The study demonstrated that journal-based recovery could restore a significant proportion of deleted files, but noted that metadata reconstruction remained unreliable without direct inode traversal. Sweeney et al. [2] provided foundational insight into XFS scalability and internal block management, establishing the theoretical basis for understand-ing how deletion marks space as available without immediately erasing underlying data. Lee et al. [3] investigated XFS file recovery tools and demonstrated the feasibility of signature-based carving within XFS block regions. However, their approach lacked automation and integration with metadata recovery pipelines.

➢ *Btrfs Recovery and Copy-on-Write Challenges*

Btrfs presents a qualitatively different recovery problem compared to journaling filesystems. Its copy-on-write archi-tecture means that modified data blocks are written to new locations rather than overwriting existing ones, which can preserve older file versions but simultaneously fragments the recovery landscape across multiple tree structures and snapshot references. Hilgert et al. [4] extended the Sleuth Kit forensic framework to support pooled storage filesystems and highlighted the difficulties of analysing Btrfs B-tree structures during forensic investigation. Their work underscored the need for filesystem-aware traversal algorithms capable of navigating snapshot subvolumes and orphaned leaf nodes. Pesic et al. [5] benchmarked Ext4, XFS, and Btrfs under virtualised environments and noted that Btrfs recovery rates declined sharply when snapshots were absent, reinforcing the dependency of Btrfs recovery on subvolume integrity.

➢ *Metadata Recovery and Forensic Reporting*

A persistent gap in existing recovery literature is the treatment of file metadata as a secondary concern rather than an integral component of the recovery process. Kim et al.

Demonstrated the forensic utility of journal log analysis in Ext4 for reconstructing file access timelines, but their methodology was not extended to XFS or Btrfs. Lee et al.

Proposed ExtSFR, a scalable file recovery framework for Ext-based systems, which incorporated metadata restoration as part of its pipeline. While ExtSFR achieved strong results within its target filesystem, the authors acknowledged that adapting the framework to Btrfs would require fundamental architectural changes. Jo et al. [8] and Shin et al. [9] explored forensic practices in AI speaker ecosystems, highlighting the importance of metadata preservation for constructing reliable digital evidence chains. These studies collectively reinforce the forensic significance of metadata alongside raw file data, a principle that MetaRecoverX operationalises across both XFS and Btrfs environments.

In summary, prior work has established important founda-tions in filesystem-specific recovery techniques and forensic metadata analysis. However, no existing tool or framework offers the combination of multi-filesystem support, deep file carving across sixteen or more file types, integrated metadata extraction, cryptographic verification, and automated foren-sic report generation within a single deployable application. MetaRecoverX addresses this gap directly.

## III. PROBLEM CONTEXT: FILESYSTEM FORENSICS AND RECOVERY CHALLENGES

This section characterises the technical challenges associated with recovering deleted files and metadata from XFS and Btrfs filesystems, and outlines the functional requirements that informed the design of MetaRecoverX.

➢ *Overview of the Forensic Recovery Workflow*

The recovery of deleted files from a Linux filesystem generally proceeds through a sequence of investigation stages. In the context of MetaRecoverX, this workflow is structured as follows:

- Target Selection: The investigator specifies the filesystem partition or disk image to be analysed, along with the

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr738

desired output directory for recovered files.

- Filesystem Identification: The tool detects whether the target uses XFS or Btrfs and applies the appropriate internal recovery strategy.
- File Carving: Raw block data is scanned for magic byte signatures corresponding to supported file types, including JPEG, PNG, PDF, MP3, MP4, DOCX, and others.
- Metadata Extraction: For each recovered file, associated metadata is extracted, encompassing timestamps, file size, permissions, EXIF attributes for images, and embedded document properties for PDFs and Office files.
- Integrity Verification: SHA-256 cryptographic hashes are computed for all recovered files to confirm data integrity and support chain-of-custody documentation.
- Keyword Search: Recovered file contents are optionally searched for user-specified keywords to assist targeted investigation.
- Report Generation: A structured forensic report is produced in PDF and CSV formats, consolidating all recovery findings, metadata records, hash values, and timeline information.

➢ *Technical Challenges in XFS and Btrfs Recovery*

The recovery process for XFS and Btrfs filesystems involves distinct technical obstacles that differentiate them from simpler filesystems such as FAT32 or Ext2.

- Journal Overwriting in XFS: XFS maintains a circular journal that is continuously reused. Once journal entries referencing deleted inodes are overwritten, inode-level metadata becomes unrecoverable through journal analysis alone, necessitating signature-based carving as a fallback.
- Copy-on-Write Fragmentation in Btrfs: The copy-on-write mechanism distributes file data across non-contiguous block regions, making it difficult to reconstruct complete files through linear block scanning without B-tree traversal.
- Snapshot Complexity: Btrfs snapshots maintain independent references to shared data blocks. Deleted files may persist within snapshot subvolumes even after removal from the primary filesystem tree, requiring snapshot-aware recovery logic.
- Metadata Fragmentation: Both XFS and Btrfs store metadata in distributed structures rather than contiguous regions, making complete metadata reconstruction dependent on successful navigation of these internal data structures.
- Absence of Integrated Tooling: No existing open-source tool combines carving, metadata recovery, hash verification, and reporting for both XFS and Btrfs within a single graphical and command-line application.

➢ *System Requirements and Design Principles*

The design of MetaRecoverX was guided by a set of core requirements identified through analysis of existing forensic workflows and the limitations of prior recovery tools:

- Multi-filesystem support: The tool must operate effectively on both XFS and Btrfs partitions without requiring filesystem-specific configuration by the end user.
- Broad file type coverage: Carving support must span at least sixteen common file types identifiable through magic byte signatures.
- Complete metadata recovery: The tool must extract and preserve file timestamps, permissions, EXIF data, and embedded document metadata alongside recovered file content.
- Cryptographic integrity assurance: SHA-256 hashing must be applied to all recovered files to support forensic admissibility.
- Dual interface accessibility: Both a command-line interface for scripted forensic workflows and a graphical interface for interactive investigation must be provided.
- Automated forensic reporting: The system must generate structured PDF and CSV reports without requiring manual compilation by the investigator.
- Practical deployability: The tool must run on standard Ubuntu Linux without requiring specialised hardware or proprietary dependencies.

## IV. PROPOSED METARECOVERX FRAMEWORK

This section presents the architecture and component design of MetaRecoverX, a purpose-built Python forensic framework for recovering deleted files and metadata from XFS and Btrfs filesystems. The framework is structured around five core modules that operate in a coordinated pipeline, from initial filesystem analysis through to final report generation. The system is designed to be modular, extensible, and deployable on standard Ubuntu Linux environments without requiring proprietary dependencies.

➢ *System Architecture Overview*

MetaRecoverX follows a layered architecture in which each module handles a distinct stage of the forensic recovery process. The five primary layers are the filesystem interface layer, the file carving engine, the metadata extraction module, the integrity and search layer, and the reporting and output module. A dual interface layer comprising both a PyQt6 graphical user interface and an argparse-driven command-line interface sits above the core pipeline, allowing investigators to interact with the system according to their operational requirements.

The tool is implemented entirely in Python and leverages standard libraries including os, struct, hashlib, and subprocess, alongside third-party packages such as Pillow for image metadata, python-docx and PyPDF2 for document attribute extraction, and reportlab for PDF report generation.

➢ *Filesystem Interface and Target Identification*

The filesystem interface layer accepts a target partition path or disk image as input and determines the underlying filesystem type using subprocess calls to standard Linux utilities. For XFS targets, the tool invokes xfs_db in read-only mode to inspect inode allocation records and identify blocks associated with deleted file entries. For Btrfs targets, btrfs-progs utilities are used to enumerate subvolumes and traverse

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr738

B-tree leaf nodes in search of orphaned file references.

Where direct inode or B-tree traversal does not yield recoverable records, the tool transitions to raw block scanning, treating the partition as a binary stream and applying the file carving engine to identify recoverable file signatures across the entire addressable space.

➢ *File Carving Engine*

The file carving engine constitutes the primary recovery mechanism of MetaRecoverX. It operates by scanning the binary content of the target partition for known file header and footer signatures, commonly referred to as magic bytes. Each supported file type is associated with a header signature that uniquely identifies the start of a file and, where applicable, a footer signature that marks its end.

MetaRecoverX supports carving across sixteen or more file types. Each supported format is identified by its unique header magic bytes: JPEG (FFD8FF), PNG (89504E47), PDF (25504446), MP3 (494433), MP4 (66747970), GIF (474946), BMP (424D), DOCX and XLSX (ZIP-based Office

Open XML formats identified via 504B0304), WAV and AVI (RIFF-based formats via 52494646), ZIP (504B0304), and RAR (52617221), among others.

Upon locating a valid header, the engine extracts the corresponding byte sequence up to the identified footer or a configurable maximum file size boundary, writes the recovered content to the designated output directory, and logs the recovery event with block offset information for forensic reference.

➢ *Metadata Extraction Module*

Following successful file carving, the metadata extraction module processes each recovered file to extract associated at-tribute information. The module operates across three categories of metadata:

- Filesystem Metadata: File creation time, last access time, last modification time, file size in bytes, and Unix permission flags are extracted using os.stat() calls on recovered file objects.
- EXIF Metadata: For recovered image files in JPEG and TIFF formats, the Pillow library is used to extract EXIF attribute dictionaries, including camera model, capture timestamp, GPS coordinates where present, image dimensions, and colour space information.
- Document Metadata: For recovered PDF files, PyPDF2 extracts document information fields including author, title, creation date, and producer. For recovered DOCX and XLSX files, python-docx and openpyxl extract core prop-erties including creator, last modified by, and document revision count.

All extracted metadata is stored in a structured internal record associated with each recovered file, which is subsequently passed to the reporting module.

➢ *Integrity Verification and Keyword Search*

The integrity verification component computes a SHA-256 cryptographic hash for each recovered file using Python's hashlib library. Hash values are recorded in the recovery log alongside file paths and metadata records, enabling forensic investigators to verify that recovered files have not been modified after extraction and to compare recovered files against known hash databases.

An optional keyword search function allows investigators to specify one or more search terms prior to initiating recovery. After carving, the tool scans the text-readable content of recovered files for the specified keywords and flags matching files in the investigation report. This functionality is particularly relevant in legal or investigative contexts where specific document content must be located across a storage medium.

➢ *Dual Interface: CLI and PyQt6 GUI*

MetaRecoverX provides two interaction modalities to ac-commodate different operational contexts. The command-line interface, implemented using Python's argparse module, accepts parameters including the target partition path, output directory, file types to carve, and optional keyword search terms. The CLI supports scripted forensic workflows and can be integrated into automated investigation pipelines.

The graphical user interface, built using PyQt6, provides a window-based environment in which investigators can browse for target partitions, configure recovery settings through form controls, monitor recovery progress through a live status display, and initiate report generation with a single button interaction. The GUI mirrors the full functionality of the CLI, ensuring that no capability is exclusive to either interface. Recovered files are directed to a configurable output path; by default the GUI writes to the recovered_output folder under the project directory, while the CLI writes to ˜/Desktop/recovered/.

➢ *Forensic Report Generation*

Upon completion of a recovery session, MetaRecoverX automatically generates a structured forensic investigation report in both PDF and CSV formats using the reportlab library and Python's built-in csv module respectively. The PDF report includes a cover section identifying the tool version, investigation date, and target partition details, followed by a tabulated summary of all recovered files with their types, sizes, metadata values, SHA-256 hashes, and recovery status. A time-line section presents recovered files ordered by their extracted timestamps, supporting chronological reconstruction of file system activity. The CSV report mirrors this information in a machine-readable format suitable for import into spreadsheet applications or external forensic analysis platforms.

## V. EXPERIMENTAL SETUP AND METHODOLOGY

➢ *Test Environment*

All experiments were conducted on a system running Ubuntu 24.04 LTS in a dual-boot configuration alongside

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr738

Windows, with approximately eighty-five gigabytes allocated to the Linux partition. MetaRecoverX was developed and executed within a Python virtual environment located inside the project directory at MetaRecoverX/ under the user's home path, with Visual Studio Code serving as the primary development environment. System dependencies including xfsprogs and btrfs-progs were installed at the system level, while Python package dependencies were managed within the virtual environment.

Two dedicated test partitions were created for the purpose of evaluation: one formatted with XFS and one formatted with Btrfs. Each partition was populated with a structured set of test files spanning multiple categories.

➢ *Test File Dataset*

The test dataset comprised files across eight categories: plain text documents, JPEG and PNG images with embedded EXIF data, MP3 audio recordings, MP4 video files, PDF documents with embedded author metadata, DOCX word processing files, and ZIP archives. Files ranged in size from a few kilobytes to several megabytes to evaluate carving performance across different data volumes.

➢ *Deletion and Recovery Scenarios*

Three deletion scenarios were evaluated for each filesystem:

- Simple Deletion: Files were removed using the standard rm command, leaving underlying data blocks intact but marking inode entries as free.
- Overwritten Files: A subset of deleted files was followed by write operations to the same partition region, simulating partial overwriting as encountered in active filesystem use.
- Filesystem Stress: Additional file creation and deletion cycles were performed prior to recovery to introduce fragmentation and test carving robustness under realistic forensic conditions.

➢ *Evaluation Metrics*

Recovery performance was assessed using the following metrics:

- File Recovery Rate: The proportion of deleted files successfully carved and written to the output directory as a percentage of total deleted files.
- Metadata Recovery Rate: The proportion of recovered files for which complete metadata records including timestamps, permissions, and type-specific attributes were successfully extracted.
- Hash Verification Rate: The proportion of recovered files for which SHA-256 hash computation completed without error.
- Recovery Time: The elapsed time from tool invocation to completion of report generation for each deletion scenario.

## VI. RESULTS AND ANALYSIS

➢ *File and Metadata Recovery Rates*

Table I summarises the recovery performance of MetaRe-coverX across both filesystems and all three deletion scenarios.

Table 1 Metarecoverx Recovery Rate Analysis

| Recovery Metric | XFS (%) | Btrfs (%) |
|---|---|---|
| File Data Recovery | 92 | 85 |
| Metadata Recovery | 88 | 79 |
| Hash Verification Success | 98 | 97 |
| EXIF Extraction (Images) | 91 | 83 |
| Document Metadata | 87 | 76 |

XFS demonstrated consistently higher recovery rates across all metrics. The journaling mechanism of XFS preserves inode reference data for a recoverable window following deletion, which the file carving engine leveraged to locate file boundaries with greater precision. File data recovery reached ninety-two percent on XFS under simple deletion conditions, declining to approximately seventy-eight percent when overwriting was introduced. Metadata recovery on XFS reached eighty-eight percent overall, with timestamps and permissions successfully restored in nearly all non-overwritten cases.

Btrfs exhibited lower but still practically useful recovery rates. File data recovery reached eighty-five percent under simple deletion, with the copy-on-write mechanism occasionally preserving additional block copies that aided carving in cases where the primary file tree entry had been cleared. However, metadata recovery on Btrfs was more variable, reaching seventy-nine percent overall and dropping further for files whose inodes had been reassigned across snapshot boundaries. Extended attributes and document-embedded metadata were less consistently available on Btrfs compared to XFS.

SHA-256 hash computation succeeded for nearly all recov-ered files across both filesystems, with failure cases attributable exclusively to partially overwritten files where the carved content was incomplete.

➢ *Performance Across File Types*

Among the sixteen supported file types, JPEG and PNG images yielded the highest individual recovery rates on both filesystems, attributed to their well-defined and globally unique magic byte signatures. PDF files were recovered reliably when not fragmented across non-contiguous blocks. MP4 video files presented the greatest challenge due to their large size and susceptibility to fragmentation, resulting in lower complete-file recovery rates though partial content was frequently retrievable. DOCX and XLSX files, being ZIP-based container formats, were recoverable when their central

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr738

directory structures remained intact within the carved byte range.

➢ *Recovery Time Performance*

Average recovery times varied with filesystem type, file count, partition size, and the number of file types targeted. On XFS, simple deletion scenarios completed in approximately twenty-five to thirty minutes including report generation. Complex scenarios involving overwriting and fragmentation extended average recovery time to forty to fifty minutes. On Btrfs, recovery times were longer due to B-tree traversal overhead, averaging thirty-five to forty minutes for simple scenarios and up to sixty minutes for more complex conditions. Report generation added approximately two to three minutes to total session time regardless of filesystem type.

➢ *Comparison with Existing Tools*

MetaRecoverX offers a substantially broader feature set compared to individual recovery utilities such as xfs_repair, btrfs restore, and testdisk. Neither xfs_repair nor btrfs restore provides file carving, metadata extraction, EXIF recovery, keyword search, or automated reporting. Testdisk supports partition and file recovery but does not extract type-specific metadata or generate structured forensic reports. MetaRecoverX unifies these capabilities within a single application supported by both graphical and command-line interfaces, representing a meaningful advancement in accessible forensic tooling for XFS and Btrfs environments.

## VII. CONCLUSION & FUTURE WORK

This paper presented MetaRecoverX, a Python-based forensic data recovery framework designed to address the limitations of existing tools in recovering deleted files and associated metadata from XFS and Btrfs Linux filesystems. The proposed framework integrates file carving across sixteen or more file types using magic byte signatures, structured metadata extraction encompassing filesystem attributes, EXIF data, and document-embedded properties, SHA-256 cryptographic integrity verification, optional keyword search, and automated forensic report generation in PDF and CSV formats. A PyQt6 graphical user interface accompanies a fully functional command-line interface, making the tool accessible to both specialist forensic investigators and general users.

Experimental evaluation on Ubuntu 24.04 LTS demonstrated file recovery rates of ninety-two percent on XFS and eighty-five percent on Btrfs under simple deletion conditions, with metadata recovery rates of eighty-eight percent and seventy-nine percent respectively. These results confirm the practical utility of MetaRecoverX as a forensic instrument and establish it as a more comprehensive alternative to existing filesystem-specific recovery utilities.

➢ *Future Work*

Several directions exist for extending the capabilities of MetaRecoverX in subsequent research and development:

- Extending filesystem support to include Ext4, F2FS, and NTFS partitions to broaden applicability across mixed operating system environments.
- Incorporating machine learning-based file fragment classification to improve recovery rates for heavily fragmented files, particularly in Btrfs environments.
- Adding support for encrypted partition analysis using user-supplied decryption credentials, relevant to forensic investigation of secured storage devices.
- Developing a network forensics mode enabling MetaRecoverX to analyse remotely mounted filesystem images over secure connections.
- Integrating timeline visualisation directly within the PyQt6 GUI, enabling investigators to explore recovery timelines interactively rather than through static report documents.
- Conducting large-scale evaluation across enterprise-grade storage configurations including RAID arrays and logical volume manager environments.

MetaRecoverX provides a practical and extensible foundation for advancing open-source forensic tooling for modern Linux filesystems, with demonstrated capability across diverse file types, deletion scenarios, and metadata categories.

## REFERENCES

[1]. Z. Wang, “Research of Data Storage Mode and Recovery Method Based on XFS File System,” *Research Institute of Electronic Science and Technology, University of Electronic Science and Technology of China*, Chengdu, China, 2020.

[2]. A. Sweeney, D. Doucette, and W. Hu, “Scalability in the XFS File System,” *USENIX Annual Technical Conference*, 1996.

[3]. C. Lee, A. Majore, C. Sekie, and T. Shon, “XFS File System and File Recovery Tools,” *International Journal of Smart Home*, vol. 7, no. 1, Jan. 2013.

[4]. J.-N. Hilgert, M. Lambertz, and D. Plohmann, “Extending The Sleuth Kit and its Underlying Model for Pooled Storage File System Forensic Analysis,” *Digital Investigation*, vol. 22, pp. S76–S85, 2017.

[5]. D. Pesic, B. Djordjevic, and V. Timcenko, “Competition of Virtualized Ext4, XFS, and Btrfs File Systems under Type-2 Hypervisor,” *Proceedings of the IEEE Conference*, 2021.

[6]. D. Kim, J. Park, K.-G. Lee, and S. Lee, “Forensic Analysis of Android Phone Using Ext4 File System Journal Log,” in *Future Information Technology, Application, and Service*, Springer, Dordrecht, 2012, pp. 435–446.

[7]. S. Lee, W. Jo, S. Eo, and T. Shon, “ExtSFR: Scalable File Recovery Framework Based on an Ext File System,” *Multimedia Tools and Applications*, vol. 79, pp. 16093–16111, 2019.

[8]. W. Jo, Y. Shin, H. Kim, D. Yoo, D. Kim, C. Kang, J. Jin, J. Oh, B. Na, and T. Shon, “Digital Forensic Practices and Methodologies for AI Speaker Ecosystems,” *Digital Investigation*, vol. 29, pp. S80–S93, 2019.

[9]. Y. Shin, H. Kim, S. Kim, D. Yoo, W. Jo, and T. Shon, “Certificate Injection-Based Encrypted Traffic

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr738

Forensics in AI Speaker Ecosystem," *Forensic Science International: Digital Investigation*, vol. 33, p. 301010, 2020.
[10]. B. Carrier, *File System Forensic Analysis*. Addison-Wesley Professional, 2005.
[11]. G. C. Kessler, "File Signatures Table," *Gary Kessler Forensics*, 2010. [Online]. Available: https://www.garykessler.net/library/file_sigs.html
[12]. A. Pal and N. Memon, "The Evolution of File Carving," *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 59–71, 2009.
[13]. S. Garfinkel, "Carving Contiguous and Fragmented Files with Fast Object Validation," *Digital Investigation*, vol. 4, pp. 2–12, 2007.
[14]. G. G. Richard and V. Roussev, "Scalpel: A Frugal, High Performance File Carver," *DFRWS Annual Conference*, 2005.
[15]. J. R. Lyle and M. Wozar, "Issues with Imaging Drives Containing Faulty Sectors," *Digital Investigation*, vol. 3, pp. 13–18, 2006.
[16]. A. Hoog, *Android Forensics: Investigation, Analysis and Mobile Security for Google Android*. Syngress, 2011.
[17]. W. McKinney, *Python for Data Analysis*, 2nd ed. O'Reilly Media, 2018.
[18]. D. M. Beazley, *Python Essential Reference*, 4th ed. Addison-Wesley Professional, 2009.
[19]. "Ext4 and XFS File System Forensic Framework Based on TSK," *Journal of Digital Forensics, Security and Law*, 2020.
[20]. ReportLab Inc., "ReportLab PDF Library User Guide," 2023. [Online]. Available: https://www.reportlab.com/docs/reportlab-userguide.pdf