AutoDFBench 1.0: a benchmarking framework for digital forensic tool testing and generated code evaluation

Akila Wickramasekara, Tharusha Mihiranga, Aruna Withanage, Buddhima Weerasinghe, Frank Breitinger, John Sheppard, Mark Scanlon

Angaben zur Veröffentlichung / Publication details:

Wickramasekara, Akila, Tharusha Mihiranga, Aruna Withanage, Buddhima Weerasinghe, Frank Breitinger, John Sheppard, and Mark Scanlon. 2026. "AutoDFBench 1.0: a benchmarking framework for digital forensic tool testing and generated code evaluation." *Forensic Science International: Digital Investigation* 56 (Supplement): 302055. https://doi.org/10.1016/j.fsidi.2026.302055.

Forensic Science International: Digital Investigation 56 (2026) 302055

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

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

DFRWS EU 2026 - Selected Papers from the 13th Annual Digital Forensics Research Conference Europe

AutoDFBench 1.0: A benchmarking framework for digital forensic tool testing and generated code evaluation

Akila Wickramasekara [a,*], Tharusha Mihiranga [b], Aruna Withanage [c], Buddhima Weerasinghe [d], Frank Breitinger [c], John Sheppard [b], Mark Scanlon [a]

[a] Forensics and Security Research Group, School of Computer Science, University College Dublin, Belfield, Dublin 4, Ireland
[b] Forensics and Security Research Group, Department of Computing & Mathematics, South East Technological University, Ireland
[c] Cybersecurity, University of Augsburg, Augsburg, Germany
[d] School of Computer Science, University of Birmingham, Birmingham, United Kingdom

ARTICLE INFO

Keywords:
Digital forensics
Tool testing and validation
Generated code validation
Benchmark
NIST computer forensics tool testing program (CFTT)

ABSTRACT

The National Institute of Standards and Technology (NIST) Computer Forensic Tool Testing (CFTT) programme has become the *de facto* standard for providing digital forensic tool testing and validation. However to date, no comprehensive framework exists to automate benchmarking across the diverse forensic tasks included in the programme. This gap results in inconsistent validation, challenges in comparing tools, and limited validation reproducibility. This paper introduces AutoDFBench 1.0, a modular benchmarking framework that supports the evaluation of both conventional DF tools and scripts, as well as AI-generated code and agentic approaches. The framework integrates five areas defined by the CFTT programme: string search, deleted file recovery, file carving, Windows registry recovery, and SQLite data recovery. AutoDFBench 1.0 includes ground truth data comprising of 63 test cases and 10,968 unique test scenarios, and execute evaluations through a RESTful API that produces structured JSON outputs with standardised metrics, including precision, recall, and F1 score for each test case, and the average of these F1 scores becomes the *AutoDFBench Score*. The benchmarking framework is validated against CFTT's datasets. The framework enables fair and reproducible comparison across tools and forensic scripts, establishing the first unified, automated, and extensible benchmarking framework for digital forensic tool testing and validation. AutoDFBench 1.0 supports tool vendors, researchers, practitioners, and standardisation bodies by facilitating transparent, reproducible, and comparable assessments of DF technologies.

1. Introduction

The rapid growth of digital crimes, accelerated by the emergence of generative Artificial Intelligence (AI) technologies, has significantly increased the workload and complexity faced by digital forensic (DF) investigators (Wickramasekara et al., 2025a; Costantini et al., 2019). To address these challenges, there is a growing demand for robust, accurate, and consistent digital forensic tools that can ensure reliability and reproducibility in evidence analysis. However, one of the most persistent issues in this domain remains the lack of standardisation in tool validation and evaluation methodologies (Casino et al., 2022).

Recent advancements in large language models (LLMs) and autonomous AI agents have shown great potential in enhancing the efficiency and productivity of digital investigations (Wickramasekara and Scanlon, 2024; Scanlon et al., 2023; Khatiwala et al., 2025). These systems can generate digital forensic scripts, automate repetitive tasks, and assist in interpreting complex data. Nevertheless, despite their promise and growing interest from the digital forensic research community (Breitinger et al., 2024), there is still no appropriate or standardised mechanism to validate and benchmark the accuracy, reliability, and forensic soundness of AI-generated source code and scripts.

This research addresses that gap by focusing on the significant enhancement and expansion of a comprehensive unified benchmarking framework for evaluating both conventional and AI-generated digital forensic tools and scripts, AutoDFBench. AutoDFBench builds upon the principles of the National Institute of Standards and Technology (NIST) Computer Forensics Tool Testing (CFTT) programme and provides an automated, extensible, and standardised mechanism for reproducibly evaluating forensic tools across multiple investigation scenarios. This paper advances the framework from an initial proof-of-concept to a 1.0 release version, covering the breadth of CFTT software test suites needed for computer forensic software tool testing and validation.

* Corresponding author.

https://doi.org/10.1016/j.fsidi.2026.302055

Available online 24 March 2026

This work makes the following contributions:

1. Enhance and expand the AutoDFBench framework through the incorporation of an improved API layer, a new CSV input layer, and the expansion of the set of test suites from one to five. The framework is released open source under the Apache 2.0 License via the project's GitHub repository (https://github.com/akila-UCD/AutoDFBench).
2. The preparation and integration of comprehensive ground truth datasets derived from the NIST CFTT programme, encompassing 63 test cases and 10,968 unique test scenarios across five test suites.
3. The experimental evaluation of the framework using NIST provided datasets demonstrates its reliability, accuracy, and consistency in benchmarking digital forensic tools and AI-generated code.

2. Computer Forensics Tool Testing programme

The CFTT programme[1] provides a structured methodology for evaluating digital forensic tools, aiming to ensure reliable, repeatable, and standardised results that withstand legal scrutiny. It includes formal test specifications and publishes tool testing results that assess performance across tasks, including string search, file carving, deleted file recovery, registry analysis, database extraction, write blocking, cloud data extraction, media preparation, and mobile devices. These suites are widely recognised as *de facto* standards for independent DF tool validation. In this work, CFTT underpins AutoDFBench 1.0, supplying the benchmark datasets and specifications required for automated, reproducible, and comparable evaluations of forensic tools and scripts.

2.1. Forensic string search (FSS)

In FSS, NIST emphasises two main requirements for a string search tool. One requirement is that the tool returns exact matches based on the given query, and the other is that it should support at least one character representation for searching. To address these requirements, forensic string search is accompanied by ten primary test cases defined by the NIST CFTT programme, which are explained in Table 1.

Table 1
Forensic string search test cases.

Test Case	Description
FT-SS-01	Test the ability to identify a plain ASCII string consisting of a single word.
FT-SS-02	Assess whether a tool can locate an ASCII string regardless of upper or lowercase variations.
FT-SS-03	Verify that a tool matches whole words exactly and does not detect substrings, and ignores case.
FT-SS-04	Evaluate searches that require two different terms to be present within the same file.
FT-SS-05	Check if a tool can detect the presence of at least one string from a given set of words.
FT-SS-06	Confirm whether a search can identify a string while excluding content containing another specified term.
FT-SS-07	Determine support for searching text in non-English scripts, including Asian characters, Hangul, Kana, Cyrillic, Latin with accents, and right-to-left languages.
FT-SS-08	Assess built-in search functions for structured data types such as email addresses, phone numbers, or identification numbers.
FT-SS-09	Measure performance in scenarios such as formatted documents, fragmented files, inaccessible storage areas, NTFS metadata, substrings in file names, and word stemming.
FT-SS-10	Examine the handling of pattern-based searches using regular expressions, including hexadecimal matching and simple character sets.

[1] https://www.nist.gov/itl/ssd/software-quality-group/computer-forensics-tool-testing-program-cftt.

These ten test cases are subdivided into sub-test cases, as each test case consists of different string values, file statuses (active, deleted, and unallocated), and the disk operating system.

2.2. Deleted file recovery

Recovery of file system metadata is a crucial task in digital forensics, and tools that support this exercise are subject to mandatory requirements defined by the NIST CFTT programme. These tools should correctly identify deleted entries and deleted file system metadata. They should report errors and be able to construct recovered objects from allocated or non-allocated space, consisting of data blocks from deleted blocks.

Considering these requirements, NIST defined 14 mandatory test cases and three optional test cases (DFR-15, DFR-16, DFR-17). Table 2 summarises the mandatory test cases.

2.3. File carving

File carving is an essential technique in digital forensics, reconstructing files from internal structures without original metadata, and it is most frequently used to recover artefacts from unallocated space (Pal and Memon, 2009). To evaluate the file carving capabilities in DF tools, NIST created a set of test cases covering a variety of scenarios, including contiguous, padding-added, and byte-shifted files. These test cases are detailed in Table 3. Along with these test cases, they also provided test disk image files containing example data related to each test scenario.

2.4. Windows Registry Recovery

The Windows Registry is a central database that records system and configuration settings, user activity, and application settings, making it a critical source of forensic evidence (Singh et al., 2020). NIST provided three main requirements for a Windows registry recovery tool. One tool should support one or more hive files and a disk image with a Windows partition. The second requirement is that the tool should be able to notify users about abnormal information in the hive registry files. Lastly, it should be able to interpret the registry objects. With these requirements, NIST defined 15 core test cases and 25 optional test cases to evaluate a registry recovery tool. Table 4 summarises the core test cases for Windows Registry extraction. Normal Registry (NR) test cases focus on the structure of the hive file, while Corrupted Registry (CR) test cases evaluate the tool's ability to recover corrupted hive files. Manipulated Registry (MR) test cases assess the tool's reaction to manipulated hive files.

Table 2
Deleted file recovery core test cases.

Test Case	Description
DFR-01	Evaluates the recovery of a single non-fragmented file.
DFR-02	Assesses the recovery of a file consisting of two fragments.
DFR-03	Examines the recovery of a file that has multiple fragments.
DFR-04	Tests recovery of several non-fragmented files with non-Latin character filenames.
DFR-05	Evaluates the recovery of two fragmented files.
DFR-06	Check the recoverability of a single large file.
DFR-07	Tests recovery of one file that has been overwritten.
DFR-08	Examines the recovery of multiple overwritten files.
DFR-09	Evaluates recovery when a large number of files are deleted without overwriting.
DFR-10	Assesses recovery of a large number of files that were deleted with partial overwriting.
DFR-11	Tests recovery of files deleted from a single directory.
DFR-12	Assesses recovery of files deleted across multiple directories.
DFR-13	Checks recovery of chaotic file system activity.
DFR-14	Evaluates recovery of non-standard or other file system objects.

Table 3
File carving test cases.

Test Case	Description
FC-01	Evaluates recovery of sector-aligned contiguous files with no padding between files.
FC-02	Assesses recovery of byte alligned contiguous files without padding between files.
FC-03	Tests recovery of contiguous files with padding added between files.
FC-04	Evaluates recovery of files with fragmented order.
FC-05	Check the recovery of files with out of order fragments.
FC-06	Assesses recovery of an intermixed fragmented file.
FC-07	Tests recovery of partial or incomplete files.

Table 4
Windows registry core feature test cases.

Test Case	Description
NR-01	Verify support for processing different data types.
NR-02	Evaluate registry file with simple tree structure
NR-03	Evaluate registry trees with 512 levels or more levels.
NR-04	Assess files which have long key names.(255 or more bytes)
NR-05	Assess files which have long value names. (16,383 or more bytes)
NR-06	Evaluate handling file with large data. (>16,344 bytes)
NR-07	Verify correct handling of non-ASCII characters.
NR-08	Test the scenarios of unusual names for keys and values.
CR-01 to CR-05	Validate tool behaviour when processing corrupted or wiped hive files.
MR-01 to MR-15	Assess resilience against manipulated hive files (e.g., hidden keys, hidden values, invalid data sizes, ambiguous encodings).

2.5. SQLite data recovery

SQLite databases are widely used across operating systems, applications, and mobile devices, making them an essential source of digital forensic evidence. Data recovery from SQLite is particularly challenging because deleted or updated rows may persist in unallocated pages, freelists, or database journaling files such as the `journal` or Write-Ahead Log (WAL) (Meng and Baier, 2019).

To provide a standard for evaluating tool reliability, the NIST CFTT programme defines five core features that are required of a SQLite recovery tool. These requirements include no file modification for the analysed file, database configuration recoverability, database schema recoverability, table content recoverability, and the recoverability of the source of data elements.

To cater to these requirements, NIST defined four test cases, which are summarised in Table 5.

2.6. Motivation for this work

Existing validation efforts, such as those provided through the NIST CFTT programme, have laid important groundwork for standardising forensic tool testing. However, these results are typically published as descriptive classifications, which makes it challenging to compare tools directly or reproduce evaluations consistently. Furthermore, most evaluations require significant manual effort and lack an automated mechanism to generate precise performance metrics.

Table 5
SQLite data recovery core test cases.

Test Case	Description
SFT-01	Verifies that the tool correctly recovers page size, journal mode, number of pages and text encoding.
SFT-02	Ensures that the tool reports the complete schema, including table listings, column names, and row information for every table.
SFT-03	Checks that the tool recovers and reports all rows, including deleted or updated entries.
SFT-04	Verifies that the tool reports the source file name for all recovered data elements.

This work is motivated by the need for a unified and automated benchmarking framework that can provide clear, quantifiable measures, such as precision, recall, and F1 score, across diverse forensic scenarios. By offering reproducible, metrics-driven evaluation, AutoDFBench 1.0 enables a more transparent comparison of tools and supports both practitioners and researchers in assessing the reliability of forensic outputs.

Keyword searching, deleted data recovery, and timeline analysis are three of the most commonly used techniques by digital forensic investigators (Hargreaves et al., 2024). Hence, AutoDFBench 1.0 focuses on implementing the corresponding and relevant NIST CFTT test suites on these topics. The String Search, Deleted File Recovery, and File Carving test suites correspond to the first two techniques – keyword searching and deleted data recovery. The corresponding test suites relevant for timeline analysis are the Windows Registry Recovery and the SQLite Data Recovery.

3. Related work

A benchmark is defined as a tool for comparative evaluation according to a specific characteristic, and a proper benchmarking mechanism should inherit key attributes such as repeatability, reproducibility, relevance, fairness, verifiability, usability, and a properly controlled dataset (Brunty, 2023; v. KistowskiJ. et al., 2015). Brunty (2023) discusses the distinction between validating a forensic method and validating a tool. At the same time, it is mentioned that the benchmark should include factors such as tool version, testing party, and the frequency of tests. Additionally, the testing lab must adhere to accreditation for tool testing, which indicates that the tool testing is conducted in a rigorous yet subjective manner.

In another study, Yates and Chi (2011) introduces a benchmark framework for mobile data acquisition. The study examined parameters such as the type of data acquired and the time required for data acquisition, which varied by device model and forensic tools. Yet, the authors conclude that they plan to include images, messages, and deleted data in the benchmark. Pan and Batten (2009) introduced a performance testing benchmark for DF tools by utilising the execution time in encryption key recovery scenarios. Moreover, the authors mentioned that the NIST CFTT and the Scientific Working Group on Digital Evidence (SWGDE) focused on evaluating the correctness of digital forensics tools. Still, some of the data are not publicly available.

The ubiquitous DF acquisition, analysis, and reporting steps necessitate investigators to rely on bespoke DF hardware and software tools. The reliability and accuracy of these tools would be 100 % in an ideal scenario, but the field does not demand 100 % accuracy 100 % of the time (Horsman, 2019). Rather, it requires accurate reporting on their reliability and constraints. Horsman (2019) highlights benchmarking challenges, including an insufficient dataset for evaluation, which focuses solely on the tool's correct functionality rather than its proper usage and error identification. Additionally, the author states that NIST provides the closest thing to a comprehensive tool for testing; however, it still does not cover all areas. Consequently, the field currently has a gap in insufficient testing standards and procedures to effectively validate a tool. Javed et al. (2022) studied the existing challenges in digital forensics, focusing on the obstacles within the resources category. They highlighted a significant gap in the unavailability of benchmarks and datasets, which hinders proper evaluations.

Cherif et al. (2026) proposes the DFIR-Metric to evaluate LLMs, which supports digital forensics investigation tasks. It comprises three modules: Multiple-Choice Questions (MCQs), capture the flag tasks, and disk analysis tasks, utilising NIST CFTT's forensic string search test suite. The authors evaluate the LLMs by categorising output as correct, skipped, syntax, and wrong, then calculating how many occurrences were accurate from the tested LLMs.

AutoDFBench has recently been introduced, which can evaluate tools and generate code from GenAI models focusing on the NIST forensic string search test suite (Wickramasekara et al., 2025b). Existing work on AutoDFBench only considers this single testing scenario, and the expansion of this framework forms part of the motivation for this work, as outlined in greater detail in Section 2.6.

Despite the progress of these studies, a significant research gap remains. Existing validation frameworks often focus on narrow tool categories or rely on manual, descriptive assessments, limiting reproducibility and comparability. None provides a unified, automated, and comprehensive benchmarking of both conventional and AI-generated forensic tools and scripts. This gap directly impacts trust, transparency, and standardisation in digital forensic investigations – precisely the challenge that *AutoDFBench 1.0* seeks to address through complete, automated, score based, and reproducible benchmarking.

This work extends the AutoDFBench framework by adding additional NIST test suites, including deleted file recovery, file carving, Windows registry recovery, and SQLite recovery. It also adds an API layer, integration, and a CSV input layer – elevating the framework's versioning to *AutoDFBench 1.0*.

3.1. AI-generated code in digital forensics

Recent advances in LLMs and domain-specific autonomous agents have introduced a new paradigm in digital forensic research. These systems have the potential to automatically generate code fragments or scripts that replicate forensic functions for scenarios such as string search, file carving, deleted file recovery, etc (Scanlon et al., 2023; Wickramasekara et al., 2025b). Such generative capabilities offer opportunities for rapid prototyping, automated evidence analysis, and intelligent assistance during investigations. However, they also introduce new challenges related to the reliability, reproducibility, and forensic soundness of automatically generated code and scripts. As highlighted by Dunsin et al. (2024), although AI and machine learning techniques are increasingly integrated into digital forensics and incident response, they remain hindered by issues of data validity, interpretability, and standardised validation procedures.

4. Framework design

This section outlines the design and architecture of AutoDFBench 1.0 and explains how each component enables reproducible, metrics-driven benchmarking across diverse forensic tasks. The overall design presents the relationship between ground truth data and computed metrics, modular extensibility across test suites, and how ground truth data aligns with NIST CFTT test cases to ensure the comparability and auditability of results.

4.1. Design considerations

The framework adopts a modular architecture in which each evaluation suite operates independently while sharing a common processing layer. Such separation is essential because the evaluation criteria differ across the various forensic tasks covered, i.e., each task (or subtask) may necessitate a particular definition of *correctness* that is unique to that task. While this definition might be reusable across similar tasks, no definition is universal.

Each test suite functions as an independent module with its own parameters and scoring process, enabling new forensic domains or test categories to be incorporated without requiring any modifications to the existing architecture. The same structure also supports proficiency testing, where independently prepared datasets can be introduced as new modules to assess DF laboratories and tool performance under controlled conditions.

All modules communicate with a central MySQL database that maintains ground truth, configuration, and results to ensure auditability. As illustrated in Fig. 1, the system is organised into three layers: *Score Calculation*, *API*, and *CSV Input*. The API layer acts as a mediation interface between tool outputs and scoring logic, enforcing a consistent schema across all suites. The CSV input layer supports batch and offline evaluations, promoting reproducibility and controlled experimentation.

AutoDFBench 1.0 integrates five evaluation suites corresponding to the following DF test suites defined by the NIST CFTT programme: forensic string search, deleted file recovery, file carving, Windows Registry recovery, and SQLite data recovery. Each suite is implemented as a dedicated API endpoint with parameters specific to its test category but serves a shared scoring format that includes precision, recall, and F1 score.

4.2. Database

The database schema is structured to maintain minimal redundancy while ensuring evidential completeness. Three primary tables are `config`, `ground_truth`, and `test_results`.

The `config` table contains the configuration parameters required for each module, such as source directories, disk image paths, and environment-specific settings, thereby supporting repeatable experiments across systems.

The `ground_truth` table serves as the reference dataset for all evaluations. Each record includes the test case identifier, relevant attributes, and classification fields. The `type` column differentiates between active, deleted, or unallocated files; `os` identifies the operating system context; and `cftt_task` specifies one of the evaluation domains (`string_search`, `deleted_file_recovery`, `file_carving`, `windows_registry`, or `sqlite`). For deleted file recovery, additional attributes such as `file_name`, `size`, and multiple timestamps (`access_time_stamp`, `modify_time_stamp`, `change_time_stamp`, `deleted_time_stamp`) are included, along with `dfr_blocks` for recording original allocation details. For file carving tasks, a `carve_type` field distinguishes between contiguous (`contig`), non-contiguous (`non`), and fragmented (`frag`) files.

The `test_results` table acts as both a results repository and a logging table. Each entry records the test case identifier, the tested tool, counts of true positives, false positives, and the computed F1 score, thereby maintaining a complete audit trail of all evaluations and enabling quantitative comparisons across forensic tools and generated code variants.

4.3. API layer

The API layer provides a standardised interface for evaluation and ensures consistency across test suites while allowing suite-specific parameters. The available endpoints are:

- `/api/v1/string-search/evaluate`
- `/api/v1/deleted-file-recovery/evaluate`
- `/api/v1/file-carving/evaluate`
- `/api/v1/windows-registry/evaluate`
- `/api/v1/sqlite-recovery/evaluate`

The string search endpoint requires file content and OS, whereas the deleted file recovery API requires the file name, file size, and MAC timestamps (Modified, Access, Created). For the file carving API, the carved file should be passed as form data. In Windows registry recovery, a converted CSV file is required as the input parameter. For the SQLite recovery API, the database name, file rows, column names, and primary key must be submitted depending on the test case.

Each endpoint also accepts the common parameter base test case, with the tool being used as an optional parameter. All responses return standard metrics, including precision, recall, and the AutoDFBench Score for the specific test case.

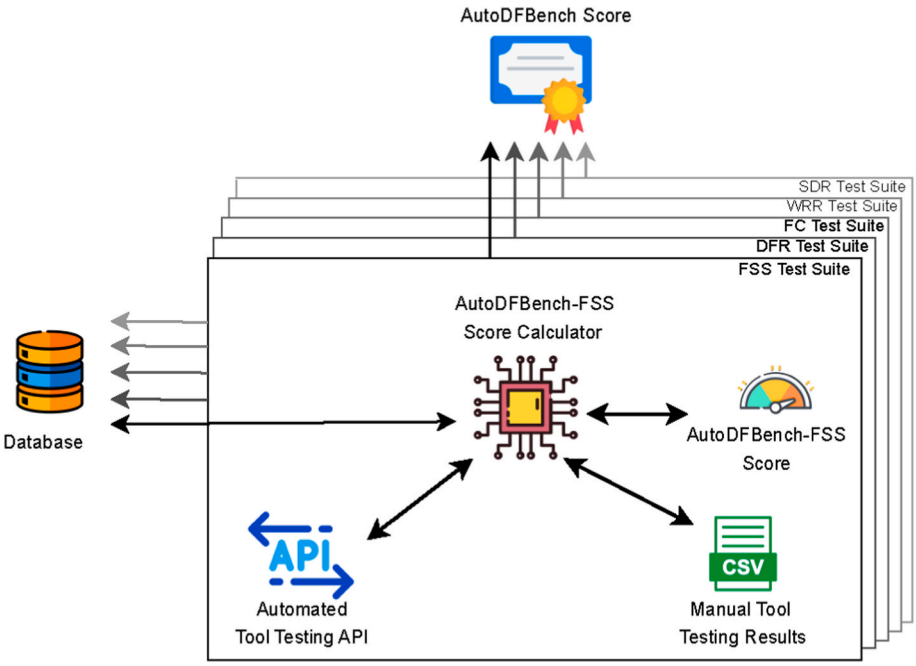

Fig. 1. Overview of the proposed framework.

4.4. CSV input layer

The CSV input layer extends functionality for batch and offline evaluations, facilitating proficiency studies and automated testing workflows. The execution script `csv_eval.py` accepts the test case name, the input CSV file path, and the output report path. The input CSV file should include all relevant test case code and the specific results obtained from the tools. This mechanism enables bulk processing of test suites without live API calls and ensures that evaluations can be reproduced precisely. The approach supports continuous integration pipelines (CI) and proficiency assessments, where deterministic output generation is critical.

4.5. Ground truth datasets

Ground truth (GT) datasets are directly derived from NIST CFTT resources and include only the attributes necessary to test the mandatory test cases.

4.5.1. String search ground truth

In string search GT, NIST has provided a unique four-digit number for each word match, where this number is concatenated with the string line of the search text. During evaluation, tool-reported string lines are mapped to these identifiers to verify exact matches. For example, in test case `FT-SS-01`, identifiers `0896`, `0898`, and `0900` indicate the lines that contain the word "DireWolf" (the keyword to be searched in these test cases).

As explained in Section 2.1, 10 main test cases are defined, which are further divided into sub-test cases representing different keywords, file types, and OS. Overall, there are 1844 forensic string search sub-test cases.

4.5.2. Deleted file recovery ground truth

For DFR test cases, file blocks, file name, file size, access time, modify time, and creation time were used as the ground truth parameters, which were provided by NIST.[2] These parameters were maintained for comparison with test cases in the score calculation.

As NIST does not publicly provide the file blocks for files in test case DFR-1, DFR-2 and DFR-3, the file blocks are calculated using the absolute sector numbers concerning the partition offset and the number of sectors that the filesystem allocates per data unit. For each absolute sector s address, the corresponding file-system block index is calculated as:

$$\text{Block}(s) = \lfloor s - PS \rfloor$$

where.

- PS is the partition start sector,

With this, the file starting block was identified, and by using the file size and sector size, it was determined how many blocks were needed for that file. With this formula file, blocks are calculated, and block ranges are stored in the ground truth table.

In the DFR test cases DFR-07, DFR-08, DFR-09, DFR-10, and DFR-13, most files lack absolute sectors, which are not included in the GT document. As a result, the GT data are not fully complete for these test cases. Despite this, for all 14 NIST test cases, there are 8147 ground truth test variations available.

4.5.3. File carving ground truth

NIST provided image files in BMP, GIF, PNG, HEIC and TIFF formats, and three disk image types were used as the ground truth data.[3] Each ground truth test case contains the image file path of each image and the contingency type (three types). In the ground truth table, the base test case is defined with a fragmented type and the image format. For example, the *carve-conti-bmp* test case is related to testing the carved BMP files taken from the disk image of contiguous files. The unique ground truth test variations for file carving, totalling 108, enable the testing of all file carving test cases as defined by NIST.

4.5.4. Windows Registry Recovery ground truth

For tool testing, NIST has provided a collection of registry files for

[2] https://cfreds-archive.nist.gov/dfr-test-images.html.

[3] https://cfreds-archive.nist.gov/filecarvingtestreports.html.

each test case. Since they only provided the hive files, it is necessary to extract the content from these files, as the NIST test cases are aligned with the testing of the contents inside the files. To do this, the `regipy` Python library was used.[4] It allows for reading keys and values inside a hive file. A set of CSV files was generated for each hive file using `regipy`, and those CSV files were used as the ground truth data to cross-check the keys and values for data sent via the API. `regipy` was selected because it is a free and open-source library that offers the required capabilities, such as reading subkeys and values. On GitHub, it has gained 261 stars and has active contributors, which verifies the solidity of the library.

This CSV file contains PATH, which maintains the hierarchical registry key namespace, TYPE maintains the data type, VALUE holds the data for a particular key, and MTIME records metadata modification tracking. These CSV files serve as the ground truth data, with the database storing the paths to these extracted files corresponding to each test case. The ground truth data consists of 49 test variations across 28 test cases.

4.5.5. SQLite recovery ground truth

To align with NIST test cases and properly evaluate them, the SQLite recovery ground truth contains a large number of parameters. These parameters include page size, journal mode, number of pages, file hash, encoding type, source file, table names, columns, and number of rows. For each test case, one or more parameters are used as validators.

For the SFT-01 test case, page size, file hash, journal mode, and number of pages act as the verification parameters. For SFT-02, table names, column names, and row ID counts are taken as the ground truth data. SFT-03 maintains the database file name and row count, and for SFT-04, it retains the source file path as the ground truth data.

Ground truth data were extracted from the NIST CFTT SDR files. To extract the necessary data from the SQLite file, *DB Browser for SQLite* was used.[5] This was chosen because it is a popular open-source tool on GitHub. This ground truth dataset includes 4 test cases and 820 variations of test data.

4.6. Score calculation

Score calculation constitutes the analytical core of the framework, transforming raw evaluation outputs into reproducible and comparable performance metrics. Each evaluation suite applies distinct scoring logic tailored to the forensic characteristics of its test category while maintaining a common set of quantitative indicators: precision, recall, and F1 score. These metrics are first computed at the subtest level, corresponding to individual CFTT cases (e.g., FT-SS-01 or DFR-01), and are then averaged to obtain a consolidated test suite score, defined as the AutoDFBench test suite score. The average of all five test suite scores forms the AutoDFBench Score, which represents the overall accuracy of a tool.

By employing this hierarchical scoring structure, the framework ensures that every evaluation, from granular subtests to complete benchmark suites, can be interpreted, replicated, and compared in a statistically consistent manner.

4.6.1. Forensic string search score

In the string search score, evaluation is based on mapping the returned string line identifiers to the ground truth. Correct matches are counted as true positives (TP), additional lines as false positives (FP), and missed identifiers as false negatives (FN). These values are used to compute precision, recall, and the F1 score for each test case, and then the Average for all the test scores as the AutoDFBench-FSS score.

4.6.2. Deleted file recovery score

Score calculation in DFR varies between test cases for overwritten (DFR-07, DFR-08, DFR-10) and non-overwritten test cases (DFR-01, DFR-02, DFR-03, DFR-05, DFR-06, DFR-09, DFR-12). For these, only deleted file blocks are considered, and it validates how many complete file blocks match the GT record. As a result, a TP is given if a recovered file matches its full set of constituent blocks. If the recovered file blocks do not contain all the blocks, it counts as a FP. If the file blocks are not relevant to a particular ground truth entry, it is counted as a FN. Subsequently, the precision, recall, and F1 score are calculated.

For the MAC times recovery test cases (DFR-01-MAC), the correct MAC times are validated, and only when all timestamps match is this considered a TP. Similarly, the file size evaluation test cases (DFR-07-SIZE, DFR-01-SIZE, DFR-011-SIZE) verify that the correct file size is captured and counted as a TP. The Latin character file test (DFR-04-CHAR) validation counts a TP when an exact file name is matched. FP is determined when a parameter does not match the ground truth, and FN is determined when the ground truth record is not coherent with the parameter record.

The F1 scores are then calculated and averaged across all the test cases to produce the AutoDFBench-DFR score.

4.6.3. File carving score

The file carving score is calculated using the same principle as deleted file recovery by determining TP, FP, and FN to compute precision, recall, and the F1 score. Each carved file is compared against the ground truth by analysing its byte sequence while also verifying that the file is decodable and can be opened, as this is essential for its use as a forensic artefact.

To identify the most relevant carved output matched to each ground truth file, the framework employs a perceptual hash (*pHash*) to select the closest visual match. Perceptual hashing generates a compact representation of an image based on its visual features, such that similar images produce hashes with a small Hamming distance, even if they differ at the pixel level due to compression or minor alterations. This robustness makes pHash well suited for validating recovered images in forensic scenarios (Samanta and Jain, 2021).

The similarity between the carved file f_c and the ground truth file f_g is measured at the byte level. If the similarity is greater than or equal to 20 % and the carved file is decodable, the recovery is counted as a TP. Otherwise, if the tool provides a carved file that does not correspond to any ground truth file, it is counted as a false positive (FP). Conversely, if a ground truth file has no corresponding carved output, or does not meet the similarity threshold, it is counted as a FN. With TP, FP and FN, the F1 score is calculated, and the Average F1 score is defined as the AutoDFBench-FC score.

The 20 % threshold was determined by carving the NIST-provided test image set, using Scalpel v1.6, where files that were empirically assessed as useable each exceeded 20 % byte similarity.

4.6.4. Windows Registry Recovery score

Similar to string search calculations, the Windows registry uses the same mechanism for text comparison. As the ground truth CSV file contains the path and value for each test case, these values are cross-checked at the string level.

When the row matches the hive file CSV, which is recovered by the tool, it is counted as TP. When the ground truth CSV contains a row that is not included in the CSV sent by the recovered tool, it is counted as a FN. Additionally, FP is counted when the recovered tool CSV transmits a row that is not present in the ground truth CSV file. With this data framework, the precision, recall, and F1 score are calculated.

4.6.5. SQLite recovery score

SQLite score calculation uses a mechanism similar to that of all other score calculators. However, for each test case, the attributes used for score calculation differ. SFT-01 checks the page size, journal mode,

[4] https://pypi.org/project/regipy/.
[5] https://github.com/sqlitebrowser/sqlitebrowser.

number of pages, file hash, and text encoding. The correct matching of these parameters counts toward the true positives (TPs) for SFT-01. If the passed parameter does not match the ground truth, it is counted as a FP, while a FN results in a zero value. This is due to the fact that SQLite recovery test cases validate a true or false condition with the given parameters from the API.

In the SFT-02 table, names, column names, and row counts are compared. In this test case, all column names, table names, and row counts must match exactly to be counted as a TP. Similar to the SFT-01, a FP is enumerated for scenarios where the ground truth data does not match. Moreover, the FN count remains zero. SFT-03 uses deleted and updated row IDs to check the ground truth data. Correctly matching row IDs are counted as a TP, while a FP counts rows that exist in the ground truth but are not included in the API request. A FN is counted for rows for which the ground truth does not have a valid record pertaining to that comparison. SFT-04 keeps the SQLite file name as the cross-checker, and if the correct file name matches, it is counted as a TP, and if not, as a FP, and again, a FN is counted as zero.

F1 scores are computed separately for each test case, and the average of each test variation is then defined as the AutoDFBench-SDR score.

5. Experimentation

To evaluate the integrity and accuracy of the framework APIs and the score calculation functionality, experiments were conducted using the publicly available test datasets provided by NIST. Since each test suite contains a distinct set of test cases, the experiments were executed individually for each suite. The framework was tested in a Conda environment, where all necessary Python libraries were installed to support the APIs for each test suite. The corresponding libraries, along with API requests and responses, are made available in the GitHub repository referenced in Section 1.

5.1. Forensic string search experimentation

The FSS test data were obtained from NIST's Federated Testing v4.0 files. These datasets include the exact string lines expected as output for each test case, together with the corresponding disk image (`*.dd`) files for both Linux and Windows environments.[6] For evaluation, the provided string lines were submitted to the framework via the API, along with the corresponding base test case name. To automate this process, all string line parameters were defined in a CSV file and mapped to each test case. A Python script was then used to execute the test cases sequentially, storing the output F1 scores in another CSV file.

5.2. Deleted file recovery experimentation

To verify the file block matching approach for the ground truth data DFR-01, DFR-02, DFR-03, and DFR-05 test cases, tests were conducted using The Sleuth Kit 3.2.2 (TSK 3.2.2).[7] For this, NIST has provided disk image files for each file system, i.e., EXT, FAT, and NTFS. Using these image files, deleted records were observed, and the file blocks for each test case were recorded. They were then tested via the framework's API. Due to the unavailability of data for ground truth, DFR-06, DFR-07, DFR-08, DFR-09, DFR-10, and DFR-013 test cases were not evaluated.

Recovery MAC times and deletion through the recycle bin were also tested as a subtest case of DFR-01. Similarly, file size evaluation scenarios were tested as subtest cases of DFR-01 and DFR-11. NIST uses these subtest cases because they utilise the same disk image files for those tests. However, the main purpose of the experimentation is to validate the logic and score calculation of the framework, enabling the testing of these test cases using data from the NIST documentation.[8]

5.3. File carving experimentation

As the scope of the experimentation in this paper is to determine the correctness of the scoring mechanism and the validity of ground truth data, the file carving scenario was tested with the same images provided by NIST that were used in the ground truth data. According to the NIST carving test cases, contiguous file scenarios are FC-01, FC-02 and FC-03, fragmented file scenarios are FC-04 and FC-05, and FC-05 represents the non-aligned cluster scenario. FC-07 was not tested, as no related NIST data is available for ground truth to check the partial file.

5.4. Windows Registry Recovery experimentation

Since the Windows registry ground truth was harvested by a Python module, validating the ground truth data and the score calculation mechanism is necessary. To perform this validation, another Python registry extraction module called `python-registry` was used. Using these Python libraries, each of the NIST-provided hive files was extracted and saved as separate CSV files. To validate the ground truth data and the API, the CSV files created by `Regipy` were also sent to the API. Output F1 scores were recorded from the CSV files generated by both tools.

For each test case, the tool-generated CSV was submitted to the Windows Registry Recovery API endpoint with the corresponding parameters, including the base test case identifier, tool designation, and job identifier. The API executed entry-level comparisons through normalised key matching, generating performance metrics including true positives, false positives, false negatives, precision, recall, and F1 scores.

5.5. SQLite recovery experimentation

The dataset was first downloaded from NIST, which contains all SQLite files covering the core test cases. These files were then loaded into the DB Browser for SQLite, and the required parameters were extracted to send to the API. In SFT-01, database headers such as *journal mode* and *page size* were obtained using the PRAGMA *page_size; PRAGMA journal_mode;*. Similarly, in SFT-02, table names, column names, and row counts were extracted. In SFT-03, the modified and deleted rows were identified using the primary keys, as explicitly documented in the SQLite test case descriptions provided by NIST. These identifiers were directly passed to the API. SFT-04 was tested by sending the exact SQL file name via the API.

6. Results and discussion

This section presents the results of the experiments conducted to validate the framework and examines the AutoDFBench Scores obtained for each test suite. The discussion highlights the accuracy, consistency, and reproducibility of the framework's evaluations.

6.1. Forensic string search results

Via the result CSV files, all sub-test case F1 scores are averaged into the main test cases, as represented in Table 6. For all test cases, the F1 score of 1 illustrates the validity of the ground truth and the correctness of the framework's score calculation.

6.2. Deleted file recovery results

Table 7 presents the results of the DFR experiments. Test cases marked with '–' denote that they were ignored for testing due to gaps in

[6] https://cfreds.nist.gov/all/NIST/StringSearch,V11.
[7] https://www.sleuthkit.org/sleuthkit/history.php.
[8] https://cfreds.nist.gov/all/NIST/DeletedFilesRecovery.

Table 6
Forensic string search test results.

Test Case	F1 score
FT-SS-01	1
FT-SS-02	1
FT-SS-03	1
FT-SS-04	1
FT-SS-05	1
FT-SS-06	1
FT-SS-07	1
FT-SS-08	1
FT-SS-09	1
FT-SS-10	1

Table 7
Deleted file recovery test results.

Test Case	F1 score with NIST Test Data	F1 score with TSK 3.2.2
DFR-01		1
DFR-01 Recovered MAC Times	1	
DFR-01 Deletion Through Recycle	1	
DFR-01 File Size	1	
DFR-02		1
DFR-03		1
DFR-04	1	
DFR-05	1	
DFR-06	–	–
DFR-07	–	–
DFR-07 File Size	1	–
DFR-08	–	–
DFR-09	–	–
DFR-10	–	–
DFR-11	1	
DFR-11 Special NTFS Situations	1	
DFR-11 File size	1	
DFR-12	1	
DFR-13	–	–
DFR-14	1	

data availability. Each of the evaluated test cases obtained an F1 score of 1, which demonstrates the validity of the score calculation logic, block calculation logic, and the ground truth data.

6.3. File carving results

As shown in Table 8, for all contiguous file scenarios (FC-01, FC-02, FC-03), the F1 score was 1, as was the case for similarly fragmented scenarios (FC-04, FC-06) and the non-aligned cluster scenario (FC-06). '–' denotes that FC-07 was not tested. Results show that all the test cases obtained an F1 score of 1, which indicates the correctness of the score calculation and the file matching logic.

6.4. Windows Registry Recovery results

Table 9 summarises the WRR experimental results. As NIST provides only the binary hive images and not the corresponding extracted registry content, the `regipy` and `python-registry` libraries were used to generate and evaluate ground truth datasets.

Table 8
File carving test results.

Test Case	F1 score
FC-01	1
FC-02	1
FC-03	1
FC-04	1
FC-05	1
FC-06	1
FC-07	–

Table 9
Windows registry recovery test results.

Test Case	F1 score for regipy	F1 score for python-registry
CR-01	–	–
CR-02	1	0.9509
CR-03	1	0.9509
CR-04	1	0.6074
CR-05	–	–
MR-01	1	0.0036
MR-02	1	0.9509
MR-03	1	0.9556
MR-04	1	0.9553
MR-05	1	0.9509
MR-06	1	0.9509
MR-07	1	0.9509
MR-08	1	0.9509
MR-09	1	0.9509
MR-10	1	0.9509
MR-11	1	0.9509
MR-12	1	0.9509
MR-13	1	0.9509
MR-14	1	0.9509
MR-15	1	0.9283
NR-01	1	0.25
NR-02	1	0.7273
NR-03	1	0.999
NR-04	1	0.7901
NR-05	1	0.7273
NR-06	1	0.4222
NR-07	1	0.4706
NR-08	–	–

For regipy, the F1 score is 1 for all the test cases, as it was used to create the ground truth data. However, for python-registry, the F1 score is near 1 in most test cases. There are still some test cases that have very low values for the data from the Python registry library. Both of these comparison values demonstrate the validity of the ground truth data and the score calculation logic.

F1 scores were computed for each test case by averaging the results of all its sub-cases. Test cases marked with '–' indicate that extraction was not possible due to dataset gaps or library processing failures, preventing the calculation of a corresponding F1 score.

6.5. SQLite data recovery results

Table 10 presents the results of the SQLite data recovery experiments. For all core test cases, the F1 score is 1. The purpose of the experiment is to validate the accuracy of the ground truth data and the score calculation function by testing the results. With the SDR test suites, it is demonstrated that the ground truth is complete and that the framework functions as expected.

7. Discussion

The results obtained from all test suites demonstrate that the framework performs as expected, producing consistent and accurate evaluation outcomes. In the Deleted file recovery and Windows registry test cases, a few ground truth datasets were incomplete or unavailable, which limited the ability to fully validate certain scenarios, as discussed

Table 10
SQLite data recovery results.

Test Case	F1 score
SFT-01	1
SFT-02	1
SFT-03	1
SFT-04	1

in Section 6.2 and Section 6.4. Despite these limitations, all other test cases across the five evaluated test suites produced the expected outputs, confirming the validity and integrity of the framework.

In contrast, test suites with deterministic outputs, such as string search, file carving, Windows registry recovery, and SQL data recovery, achieved perfect F1 scores as the match sets were precise and the ground truth data were well defined.

The benchmarks' trustworthiness is measured by the reproducibility of each test. Of course, in any digital forensics context, reproducibility is a necessity. This framework promotes reproducibility, transparency, and auditability, making it a reliable benchmark for the digital forensics community. It also adheres to a deterministic scoring logic, which is another key quality of an accurate benchmarking framework. Open-source access to the code and ground truth enables verification and extension of the framework in the future. The modular architecture is another key advantage, as it helps maintain the base architecture and facilitates the addition of more modules. On top of ground truth data and score calculation logic, this framework can be extended for practitioner proficiency testing and to evaluate the performance of future DF AI agents.

Several limitations were identified during the experimentation. The absence of complete ground truth data from NIST datasets prevents full verification of certain scenarios. This might be perceived as a drawback of the framework. However, the problem arises not because of the framework but due to the lack of accessibility to ground truth data. Additionally, the current version of the framework focuses mainly on quantitative output comparison, such as byte- or block-level matching or string-level matching. It does not yet evaluate the meaning or context of the results. For example, it cannot assess the percentage of a carved file that was carved or the number of blocks recovered from a file. As a result, it cannot distinguish between minor acceptable variations and genuine forensic errors. In the current scope of the framework, it considers only the core evaluation test cases. However, it is also vital to include optional test cases and their ground truth in future work. Additionally, the framework does not measure the processing speed or resource usage of the tested tools, meaning that performance and efficiency aspects were not considered in this benchmark. However, it can be reasonably assumed that accuracy is the most important characteristic in any digital forensic context.

Overall, these findings demonstrate that AutoDFBench 1.0 offers a reliable and reproducible approach for validating digital forensic tools and AI-generated code, while highlighting areas where further refinement and human oversight will strengthen the framework's forensic robustness.

8. Future work

The modular architecture of AutoDFBench enables seamless extension to additional NIST test suites and emerging forensic domains, including memory forensics, mobile device forensics, and cloud data extraction. Future work will focus on integrating these new test categories and expanding the database of ground truth datasets to enhance coverage and accuracy.

Furthermore, the framework demonstrates strong potential for use in proficiency testing and certification exercises, enabling forensic laboratories and practitioners to systematically assess tool performance and examiner competency under standardised conditions. In the long term, AutoDFBench could serve as the foundation for a universal benchmarking and validation platform that supports both academic research and operational digital forensic practices.

9. Conclusion

This paper presents AutoDFBench 1.0, an extended and enhanced version of AutoDFBench that serves as an automated, modular, and extensible benchmarking framework for evaluating both conventional and AI-generated digital forensic tools. Built upon the principles of the NIST CFTT programme and NIST ground truth data, the framework standardises the evaluation process across multiple forensic scenarios, including string search, file carving, deleted file recovery, Windows Registry analysis, and SQLite data recovery.

Experimental validation using NIST datasets demonstrated that the framework produces accurate, consistent, and reproducible results. By quantifying performance through precision, recall, and F1 score, AutoDFBench establishes a transparent and repeatable approach to validating forensic tools and generated code. The framework thereby contributes to improving trust, standardisation, and scientific integrity within the digital forensics community.

References

Breitinger, F., Hilgert, J.N., Hargreaves, C., Sheppard, J., Overdorf, R., Scanlon, M., 2024. DFRWS EU 10-year review and future directions in digital forensic research. Forensic Sci. Int.: Digit. Invest. 48, 301685. https://doi.org/10.1016/j.fsidi.2023.301685. https://www.sciencedirect.com/science/article/pii/S2666281723002044. DFRWS EU 2024 - Selected Papers from the 11th Annual Digital Forensics Research Conference Europe.

Brunty, J., 2023. Validation of forensic tools and methods: a primer for the digital forensics examiner. WIREs Forensic Science 5, e1474. https://doi.org/10.1002/wfs2.1474. URL: https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wfs2.1474.

Casino, F., Dasaklis, T.K., Spathoulas, G.P., Anagnostopoulos, M., Ghosal, A., Böröcz, I., Solanas, A., Conti, M., Patsakis, C., 2022. Research trends, challenges, and emerging topics in digital forensics: a review of reviews. IEEE Access 10, 25464–25493. https://doi.org/10.1109/ACCESS.2022.3154059.

Cherif, B., Bisztray, T., Dubniczky, R.A., Aldahmani, A., Alshehhi, S., Tihanyi, N., 2026. DFIR-Metric: a benchmark dataset for evaluating large language models in digital forensics and incident response. In: Taniguchi, T., Leung, C.S.A., Kozuno, T., Yoshimoto, J., Mahmud, M., Doborjeh, M., Doya, K. (Eds.), Neural Information Processing. Springer Nature Singapore, Singapore, pp. 17–31. https://doi.org/10.1007/978-981-95-4367-0_2.

Costantini, S., De Gasperis, G., Olivieri, R., 2019. Digital forensics and investigations meet artificial intelligence. Ann. Math. Artif. Intell. 86, 193–229. https://doi.org/10.1007/s10472-019-09632-y.

Dunsin, D., Ghanem, M.C., Ouazzane, K., Vassilev, V., 2024. A comprehensive analysis of the role of artificial intelligence and machine learning in modern digital forensics and incident response. Forensic Sci. Int.: Digit. Invest. 48, 301675. https://doi.org/10.1016/j.fsidi.2023.301675. URL: https://www.sciencedirect.com/science/article/pii/S2666281723001944.

Hargreaves, C., Breitinger, F., Dowthwaite, L., Webb, H., Scanlon, M., 2024. DFPulse: the 2024 digital forensic practitioner survey. Forensic Sci. Int.: Digit. Invest. 51, 301844. https://doi.org/10.1016/j.fsidi.2024.301844. URL: https://www.sciencedirect.com/science/article/pii/S2666281724001719.

Horsman, G., 2019. Tool testing and reliability issues in the field of digital forensics. Digit. Invest. 28, 163–175. https://doi.org/10.1016/j.diin.2019.01.009. URL: https://www.sciencedirect.com/science/article/pii/S1742287618303062.

Javed, A.R., Ahmed, W., Alazab, M., Jalil, Z., Kifayat, K., Gadekallu, T.R., 2022. A comprehensive survey on computer forensics: state-of-the-art, tools, techniques, challenges, and future directions. IEEE Access 10, 11065–11089. https://doi.org/10.1109/ACCESS.2022.3142508.

Khatiwala, J.P., Kwaku Ntiamoah Addai, D., Xu, W., 2025. Evaluating the reliability of digital forensic evidence discovered by large language model: a case study. In: 2025 IEEE 49th Annual Computers, Software, and Applications Conference (COMPSAC), pp. 1067–1076. https://doi.org/10.1109/COMPSAC65507.2025.00138.

Meng, C., Baier, H., 2019. bring2lite: a structural concept and tool for forensic data analysis and recovery of deleted SQLite records. Digit. Invest. 29, S31–S41. https://doi.org/10.1016/j.diin.2019.04.017. URL: https://www.sciencedirect.com/science/article/pii/S1742287619301677.

Pal, A., Memon, N., 2009. The evolution of file carving. IEEE Signal Process. Mag. 26, 59–71. https://doi.org/10.1109/MSP.2008.931081.

Pan, L., Batten, L.M., 2009. Robust performance testing for digital forensic tools. Digit. Invest. 6, 71–81. https://doi.org/10.1016/j.diin.2009.02.003. URL: https://www.sciencedirect.com/science/article/pii/S1742287609000279.

Samanta, P., Jain, S., 2021. Analysis of perceptual hashing algorithms in image manipulation detection. Procedia Comput. Sci. 185, 203–212. https://doi.org/10.1016/j.procs.2021.05.021. URL: https://www.sciencedirect.com/science/article/pii/S1877050921011030.

Scanlon, M., Breitinger, F., Hargreaves, C., Hilgert, J.N., Sheppard, J., 2023. ChatGPT for digital forensic investigation: the good, the bad, and the unknown. Forensic Sci. Int.: Digit. Invest. 46, 301609. https://doi.org/10.1016/j.fsidi.2023.301609. URL: https://www.sciencedirect.com/science/article/pii/S266628172300121X.

Singh, A., Venter, H.S., Ikuesan, A.R., 2020. Windows registry harnesser for incident response and digital forensic analysis. Aust. J. Forensic Sci. 52, 337–353. https://doi.org/10.1080/00450618.2018.1551421.

Kistowski, Jóakim.v., Arnold, J.A., Huppler, K., Lange, K.D., Henning, J.L., Cao, P., 2015. How to build a benchmark. In: Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering. Association for Computing Machinery, New York, NY, USA, pp. 333–336. https://doi.org/10.1145/2668930.2688819.

Wickramasekara, A., Scanlon, M., 2024. A framework for integrated digital forensic investigation employing AutoGen AI agents. In: 2024 12th International Symposium on Digital Forensics and Security (ISDFS), pp. 1–6. https://doi.org/10.1109/ISDFS60797.2024.10527235.

Wickramasekara, A., Breitinger, F., Scanlon, M., 2025a. Exploring the potential of large language models for improving digital forensic investigation efficiency. Forensic Sci. Int.: Digit. Invest. 52, 301859. https://doi.org/10.1016/j.fsidi.2024.301859. URL: https://www.sciencedirect.com/science/article/pii/S2666281724001860.

Wickramasekara, A., Densmore, A., Breitinger, F., Studiawan, H., Scanlon, M., 2025b. AutoDFBench: a framework for AI generated digital forensic code and tool testing and evaluation. In: Proceedings of the Digital Forensics Doctoral Symposium. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3712716.3712718.

Yates, M., Chi, H., 2011. A framework for designing benchmarks of investigating digital forensics tools for mobile devices. In: Proceedings of the 49th Annual ACM Southeast Conference. Association for Computing Machinery, New York, NY, USA, pp. 179–184. https://doi.org/10.1145/2016039.2016088.