An Abstract Model for Digital Forensic Analysis Tools - A Foundation for Systematic Error Mitigation Analysis

Christopher Hargreaves[1], Alex Nelson[2], Eoghan Casey[3]

[1]Department of Computer Science, University of Oxford, United Kingdom
[2]US National Institute of Standards and Technology
[3]University of Lausanne, Quartier Centre, 1015 Lausanne, Switzerland

Abstract

As automation within digital forensic tools becomes more advanced there is a need for a systematic approach to ensure the validity, reliability, and standardization of digital forensic results. This paper argues for intermediate output in a standardized format within digital forensic tools to allow a methodical approach to tool validation that targets errors at each stage of processing. To achieve this, a detailed process model of digital forensic analysis tools is created, extrapolating the details of the internal processes performed by monolithic forensic tools. The research deconstructs the process flow within tools and presents an 'abstract digital forensic tool', revisiting earlier abstraction layer ideas. This not only identifies the interconnected processes within tools but allows discussion of the potential error that could be introduced at each stage, and how it could potentially propagate within a tool. A demonstration, with a dataset, is also included, structurally annotated using Cyber-investigation Analysis Standard Expression (CASE).

Keywords:
digital forensics tools, digital forensics process, abstraction layers, validation, error, tool testing, CASE

1. Introduction

[1] Digital forensic science is still facing many of the problems posed in [1]. There are also repeatedly quoted challenges of volume [2] including: volume of cases themselves, devices within cases, and data on individual devices. In order to make sense of these large volumes of data, automation is necessary and always has been in some form [3]). As the field has matured, digital forensic tools have developed significantly in terms of features, enhanced with improved search, different data views (e.g., galleries, timelines, geolocation options), and more recently the integration of AI capabilities [4].

As the field has matured, the need to validate tool results has also become increasingly expected [5]. For example, the United Kingdom (UK) Forensic Science Regulator has stated "the courts have the expectation that the methods to produce the data that an expert bases their opinion on are valid"[6]. Work on error in digital forensics has included discussions of fundamental concepts [7] [8] [9], a qualitative expression of error in event reconstruction [10], and specific sources of error in digital investigations e.g., contamination [11]. Also ASTM E3016-18 'Standard Guide for Establishing Confidence in Digital and Multimedia Evidence Forensic Results by Error Mitigation Analysis'[12] includes a discussion on the need for systematic and transparent treatment of errors, including tool validation and supporting reports.

The intersection of these two trends of automation and validation creates a challenge. Tools are offering more automated results, and there is a very reasonable expectation for those results to be demonstrably correct. However, the lack of systematic testing to expose errors in digital forensic tools raises the risk that undetected errors exist in their results for current and past cases. This paper proposes a future direction to support the ongoing use of automation within digital forensics but allowing easier validation. It achieves this by revisiting abstraction layers in digital forensics, and proposing the use of a standard representation of digital forensic artifacts at each of these layers. This paper makes the following contributions:

- A model of the processes applied within a digital forensics tool.
- A thorough discussion of error at each stage of processing and how it could propagate.
- A proposal to expose the output of each abstraction layer, for improved validation of results, and avoidance of error propagation.

The remainder of the paper is structured as follows: Section 2 provides background and related work. Section 3 describes the methodology, and Section 4 provides the main body of work including the deconstruction of forensic tool processes and examples of error. Section 5 discusses recommendations and Section 6 provides a demonstration. Section 7 discusses limitations and further work, and Section 8 provides conclusions.

Email addresses: christopher.hargreaves@cs.ox.ac.uk (Christopher Hargreaves), alexander.nelson@nist.gov (Alex Nelson), eoghan.casey@unil.ch (Eoghan Casey)

[1]Certain products may be identified in this document, but such identification does not imply recommendation by the US National Institute of Standards and Technology or other agencies of the US Government, nor does it imply that the products identified are necessarily the best available for the purpose.

2. Background and Related Work

2.1. Testing and Validation

ASTM E3016-18[12] states that most digital forensic tool errors are systematic in nature rather than statistical. Therefore, many errors in digital forensic tools can be mitigated using a systematic approach. It describes the primary types of errors found in digital forensic tool implementations as *incompleteness* (INCOMP), *misinterpretation* (MISINT), and *inaccuracy* (INAC). This last type subdivides into: *existence*, i.e., do all artifacts reported as present actually exist? (INAC-EX), *alteration*, i.e., does a tool alter data in a way that changes its meaning? (INAC-ALT), *association*, i.e., for every set of items identified by a given tool, is each item truly part of that set? (INAC-AS), and *corruption*, i.e., does the forensic tool detect and compensate for missing and corrupted data? (INAC-COR)[2].

If we consider current approaches for assessing the correctness of results in digital forensics, there are several approaches. Manufacturers of tools perform their own testing with datasets, likely adopting best practices in software engineering such as test driven development, etc. Also within the community, tools will be tested for specific results against known data sets with ground truth data, e.g., Digital Corpora [13], Computer Forensic Reference DataSet (CFReDS) [14], although the "production of these disk images is a time consuming and, therefore, an expensive process"[15]. However, because of the number of processing steps required to translate from raw digital data to a higher-level observable trace, and because many digital forensic tools are closed source, some testing strategies necessarily adopt an approach that just checks final results are as expected against that ground truth[16], or input-output-validation[17]).

However, such tool output is the result of applying an opaque sequence of many processes within a tool. Furthermore, the UK Forensic Science Regulator has stated that there is an expectation to "select data that will robustly test the method and tool"[6], and while some datasets are specifically designed to identify errors within specific abstraction layers, e.g., the Digital Forensics Tool Testing Images project [18] and [19], most datasets do not have an approach to address the full extent of hidden problems in digital forensic tools.

Another approach often quoted is dual tool verification. Reference [20] reports that in a survey "55 % of respondents can be classed as 'regularly' deploying multiple tools in order to try and protect against tool issues." However, this method has the problem that multiple tools could be based on the same incorrect knowledge, assumptions or even code libraries [21]. It also has the problem that it is often very difficult to compare the output of different tools. Even if discrepancies are identified, it is even more difficult to isolate the reasons for differences[22], again because the output being compared is the result of a series of sequential, cumulative processes applied to the raw data.

2.1.1. Abstraction Revisited

In order to consider solutions to these problems it is necessary to revisit abstraction in digital forensics. Many of the early important concepts of abstraction in digital forensics are discussed in [8], and an example model is presented that includes the layers: *physical media*, *media management*, *file system*, *application*. Also discussed is that error can be introduced any time abstraction is introduced, and abstraction layers can be modelled as input, a rule set, followed by output and a margin of error. Other digital forensic process models are often a high-level abstraction of a digital *investigation* as a whole. These process models have an 'analysis' stage, which is rarely broken down further, and it is difficult to find an example where sufficient granularity exists that allows consideration of errors between internal tool abstraction layers.

Abstraction layers and loss are considered in [23], which goes into more detail than just high-level process models. Specifically it introduces "active" and "deleted" mappings between upper and lower layers of abstraction. Several examples are provided e.g., on block storage. It provides an example of how data change over time, and how the loss of data in a 'lower' layer means that the upper layer cannot be reconstructed. While this certainly applies to digital forensics tools, it is not extended to all processes within a digital forensic tool. Reference [24] also discusses abstraction layers and extends that work considering block devices in more detail, again considering a specific part of the digital forensic analysis process.

In terms of what detail should exist within a broader examination of the analysis stage, one reference provides an ontology of techniques in digital forensics[25], covering at a high-level analysis: interpretation, presentation, reporting, but also provides more granularity for each of these, highlighting for example *FileAnalysis* under *DataNodeSearching*, and *Event Reconstruction* and *Reverse Engineering* under *Analysis*. While it provides an extensive overview and organization of specific processes that could be applied to an investigation, it does not explore individual techniques in detail and does not consider those areas in the context of abstraction layers and error. Reference [26] explores an ontology and representation of digital forensic artifacts and creates a detailed mapping of how they are related, and describes artifacts in detail, but does not cover tool processes explicitly or dependencies. The US National Institute of Standards and Technology (NIST) also has a Computer Forensic Tools and Techniques Catalog[27], which provides a taxonomy of forensic tool functionalities (e.g., disk imaging, email parsing, hash analysis, string search). This is an extensive list, but there is little detail on how the categories interact. There is, therefore, limited recent work that has attempted to consider modern monolithic digital forensic tools in the context of abstraction layers and how the layers are connected and to explore how error could be introduced and propagate within them.

3. Methodology

This paper examines the extraction and analysis processes used in digital forensic tools and reapplies the abstraction layer concept from [8]. To this end, this paper will first deconstruct the digital forensic process, not with a typical high-level view

[2] Abbreviations created for the purposes of this paper.

of an investigation, but at a lower level, focusing on the extraction and analysis aspects within digital forensic tools. With that complete, it is then possible to consider examples as to where error can be introduced, and how it might propagate within such a system from one process to another.

In order to deconstruct the digital forensics analysis process at a useful granularity, this paper looks to digital forensic tool features to extrapolate the lower-level automated processes that are carried out. However, many, but not all, digital forensic tools are closed source and are also highly complex pieces of software with many interconnected components. Therefore, the approach chosen here is to extrapolate general features and capabilities from a set of digital forensic tools and then consider for each process what input is required and what the output would be. This allows not just a set of components to be identified but also a set of dependencies to be derived between them. This results in a vendor agnostic view of digital forensic tool processes: an abstract model for digital forensic tools.

4. Analysis: Deconstruction of Digital Forensic Process

This section describes modeling an abstract digital forensic analysis tool, capturing the processes involved and their dependencies. To do this, the feature set of Autopsy, AXIOM, and X-Ways were considered from their documentation, and a generalized set of processes extracted. Furthermore, dependencies between processes were also considered. This was performed by considering the tool features in the context of the abstraction model in [8], i.e., that a process needs a specific piece of data in order to function (input), will produce some output, and, therefore, there is a dependency.

From these results, Figure 1 was constructed. Each process is shown as a block with arrows indicating the flow through the software, which are labeled numerically, although the numbers themselves do not represent a sequence. Section 4.2 onward, provides a description of the processes, their input and output, argues the existence of a dependency, and finally describes examples of the potential error that can be introduced by a particular stage of processing.

Note that there are two entry points into the diagram: (1) a physical storage acquisition, which is typical of computer based investigations (and some mobile methods) and (2) a 'set of files' which may be the case in some computer investigations, but is certainly common within other mobile investigations, e.g., iPhone backup-based acquisition.

4.1. Overview

The subsections that follow describe the processes identified and shown in Figure 1. Many are standard digital forensic processes, but all are described for completeness, and to highlight potential error and precisely explain the relationships with other processes. Each error described is tagged according to the ASTM E3016-18 error types discussed in Section 2. There are some topic omissions, for example case management and bookmarking features are not discussed here since they are considered more 'administrative' rather than core analysis functionality for the purpose of error identification, although error could occur within these areas, too, and should be addressed in further work.

4.2. Parse Image Format

Description: This applies when the input to the tool is an image of physical storage media, which is a likely format for computer-based investigations. This might be a transparent layer, where a dd image is supplied, or it may require processing of the EWF [28] or AFF [29] formats.

Dependencies: Dependency *d1*: Only data captured by the earlier acquisition stage can be processed. Therefore any errors introduced by the acquisition tool, e.g., an incorrect read of a sector from the source media, or an examiner failing to take into account disk HPAs, DCOs[30] or remapped sectors[31] etc. will result in data being incorrect or missing within the examined image.

Potential Error Introduced at this Stage: Image format parsing could fail to present all blocks from within a forensic container image in their 'flat' (dd) representation (INCOMP), or present incorrect data within sectors (INAC-ALT). Alternatively it could present incorrect forensic image metadata (INAC-ALT). Some imaging tools include "maps" to record when disk regions were not recovered, mitigating INCOMP issues; but failure to incorporate such a map into downstream analysis can lead to process and analysis errors from "preserving" the original faults in the copy process (INAC-COR).

4.3. Validate Disk Image

Description: This is the process of checking that the sectors accessed in the disk image are the same as recorded from original source material. This process mitigates potential error in the previous stage since the hash can be computed over the accessed data and compared and shown to be the same as the hash recorded for the original source media. However, image metadata (i.e., the details stored within the image format such as examiner, case name, time of acquisition) are not validated by this specific process. Other processes may be in place to validate the metadata, but it is a clearly separate process.

Dependencies: Dependency *d2*: if the image format is incorrectly parsed, the hash computed over the data would not match that stored in the image file, or in separate documentation. Fortunately in this case, detection of this problem is the point of this stage.

Potential Error Introduced at this Stage:

- Failure to compute hash correctly: this could result in a message indicating corrupt evidence, thus stopping or delaying further investigation (INAC-AS).
- Failure to validate hash properly: this could allow errors from earlier to propagate, either incorrect sectors (INAC-COR) or an incomplete disk image (INCOMP).
- Failure to validate metadata: this could allow details such as acquisition date to be changed (INAC-COR).
- Failure to validate against an externally stored hash: this could create a vulnerability to tampering, where the image file content and hash could be manipulated but would still self-validate (INAC-COR).

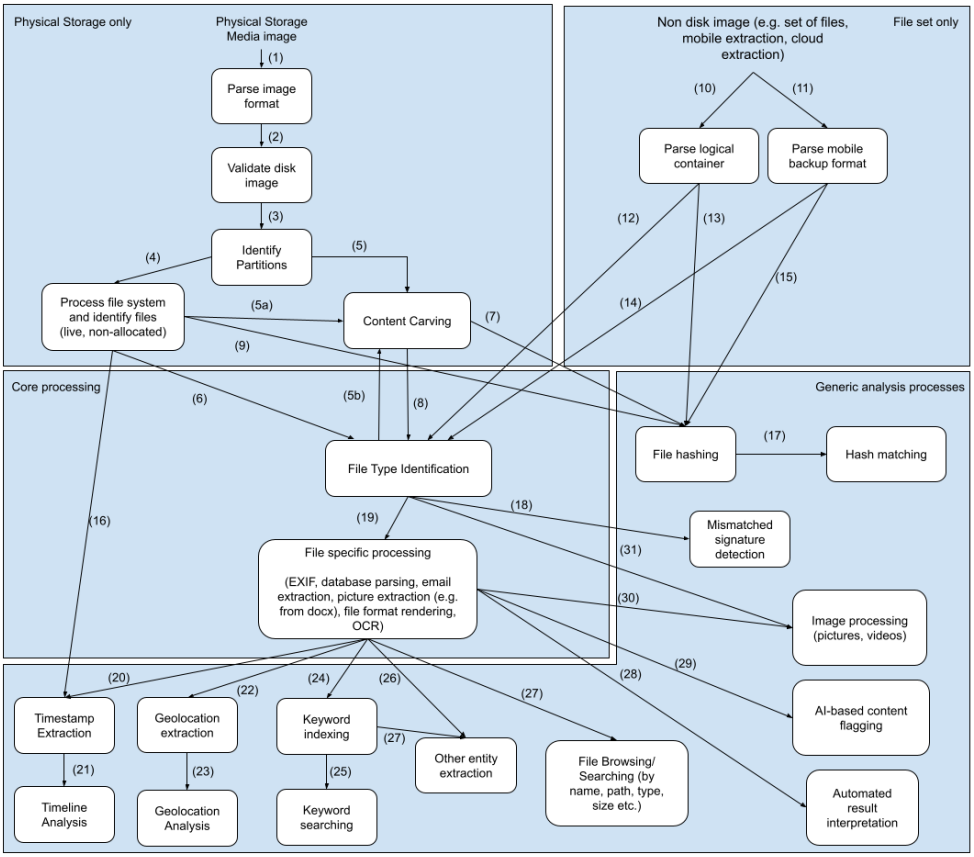

Figure 1: A representation of an abstract model for digital forensic tools indicating relationships between abstraction layers and visualizing dependencies.

4.4. Identify Partitions

Description: This should involve both parsing the partition scheme in use, e.g., GPT or MBR-based (including all the Extended Partition Tables (EPTs)), and should also include examination of the unpartitioned space and performing checks to determine if there are any deleted but still recoverable partitions, e.g., from a deleted Volume Boot Record (VBR).

Dependencies: Dependency *d3*: If data from the disk image are not validated then sectors may be missing/corrupt and prevent this partition reconstruction from being performed correctly.

Potential Error Introduced at this Stage:

- Incorrectly parsing partition table(s): if they are incorrectly processed then it could result in an incomplete or incorrect partition list. Making incorrect assumptions about sector size (e.g., 512 rather than 4096) or not parsing MBR/EPT pointers correctly could all result in missing partitions (INCOMP) or identifying ones that are not present (INAC-EX).
- Missing deleted but recoverable partitions: Even if the partition table is correctly parsed, failing to correctly search unpartitioned space for partitions that could be 'carved' out, usually from identifying a VBR, could miss entire possible partitions (INCOMP).

4.5. Processing File Systems and Identifying Files

Description: This describes the process whereby a partition is inspected, a valid file system identified, and that file system parsed according to a known (documented or reverse engineered) specification. The process also usually includes use of knowledge of the file system driver behavior to attempt to recover non-allocated files that still retain references within the file system metadata, despite being marked for reuse. This is a different process from file carving, which is discussed later.

Dependencies: Dependency *d4* means that earlier failure to correctly identify partitions (either live, or from unpartitioned space) would mean that file system processing does not occur for the content of that partition. This is an extremely significant problem since failure to identify files propagates into almost all the stages that follow, as shown in Figure 1.

Potential Error Introduced at this Stage: Many of the errors at this stage are described in detail in [32]. In general, potential errors could include:

- Failure to identify a known file system: Here either a trigger for identifying a supported file system is missed, or the tool lacks support for a file system that is known within digital forensic science. It is perhaps controversial that lack of support for a file system would be considered an error, but, in the context of trying to identify where digital forensic traces could be missed, it needs to be highlighted (INCOMP).

- Incorrect extraction of the live file set, or a subset: Here there is some error either in processing or implementing the specification; or there is a mistake or limitation in the file system specification on which the implementation is based. This results in missing files (INCOMP), additional files (INAC-EX), or inaccurate metadata or file content (INAC-ALT).
- Failure to recover one or more non-allocated but recoverable files: This means that there is file content in unallocated space that could be recovered, but the tool either does not recover it (INCOMP), or does not recover it correctly (INAC-AS). This is difficult to assess since this process is usually outside the formal specification of the file system, and different processes and tolerances for incomplete or corrupt files may be implemented by different vendors. This is, therefore, something of an abstraction error as the 'correct' value is hard to define.
- Misallocation of metadata as part of the non-allocated file recovery process: Again, since we are operating outside of the file system specification, there are examples where deleted metadata could be misallocated to recovered files (INAC-AS), or there is a presentation problem within a tool (MISINT) [32].
- Inaccurate metadata: filename, timestamp(s), permissions, etc. (INAC-ALT).
- Failure to express uncertainty in the classification of file recovery results, e.g., where content may be partly overwritten (MISINT).

4.6. Content Carving

Description: In the case of no file system metadata structures, this can mean that the file system cannot be reconstructed. Content carving does not use the file system metadata and uses file structure information ([33], [34]), e.g., header and footer, to attempt to salvage and reassemble file content. Content recovered in this way will not usually have metadata associated with it, except an extension assigned based on content type, or data formats that contain internal metadata.

Dependencies: There are three ways that content carving could be initiated. In the first case (*d5*), either unpartitioned space, or a partition containing an unsupported file system can have content carving applied. In the second case (*d5a*), within a supported file system, after live files are processed, and any non-allocated files with residual metadata extracted, the remaining unallocated space can have the content carving process applied. In the third case, the carving process is initiated after file type identification (*d5b*) and may be used directly against file objects such as the swap/pagefile. As a note, including this content carving process by default within the broader process means that error propagation may be partially mitigated for the problems: 'Missing deleted but recoverable partitions' (Section 4.4), and 'Failure to identify a known file system' within a partition (Section 4.5). This is because this should result in unexpected content being identified by the tool in this 'unpartitioned space', which may cause an investigator to challenge the automated results and look more closely at the raw data as to why a file system was not reported.

Potential Error Introduced at this Stage:

- Failure to carve salvageable content: where there is sufficient data to perform successful content carving, but a tool fails to do so. Like non-allocated file recovery earlier, this is difficult to assess since different processes and tolerances for incomplete/corrupt content may be implemented by different tools and would produce different results, but by design. This is again something of an abstraction error as the 'correct' value is hard to define (INCOMP).
- Incorrect carving of a complete file: For example a file header would be identified and content would be carved, but, due to either fragmentation or over carving, a complete file is presented that, in fact, never existed. This could be either merging the contents of multiple files, which presents a file as evidence that actually never existed, or prematurely cropping a file, which again presents an incomplete file that never existed (INAC-EX/COR).
- Incorrect reassembly of a file: In this extended example of the above case, where automated reassembly of fragments is attempted, an error here would present a complete file, but one that was assembled from fragments of separate files (INAC-EX). Either that or important chunks of the file are missing, but this is not flagged in the output (MISINT).
- Incorrect file system attribution: e.g., in [32] when a host file system contained a virtual machine disk image with guest file system of the same type recovered content could be mis-attributed to the host.

4.7. File Type Identification

Description: Later in the overall process, specific processes need to be applied to different file types. Therefore, a process must be performed that identifies the type of file. This could be via extension, although files may have been renamed, and so this is unreliable, so other means such as examining a file signature or other techniques are used [35].

Dependencies: Dependencies vary since files can be either from extracted from file system parsing (*d6*) or via carving (*d8*), or come from a logical container (*d12*) or mobile phone extraction (*d14*). This means that if files are not identified in any of these processes, fewer files can have their type identified. As the diagram shows, this has significant impact on the "file specific processing stage" (*d19*).

Potential Error Introduced at this Stage:

- Incorrect assignment of a file type: incorrectly examining the header and assigning an incorrect type, for example assigning ZIP to a DOCX file as a result of limited processing (INAC-AS). This may also occur if a file header is used by multiple file types. In general, capturing uncertainty is unusual in current forensic tools (MISINT).
- Missing assignment of a file type: reporting a file as unknown when it has a known type within the digital forensic science body of knowledge (INCOMP, INAC-AS).
- Missing recognition of multiple file types: reporting a file as a limited set of types, or a single type, when it can be interpreted as having more than one (INCOMP), such as a GIF and Java Jar [36].

4.8. File Hashing

Description: Here cryptographic hashing can be applied to files, results stored, and used later as part of a matching process.

Dependencies: Since we need the files' content in order to read and hash the data within them, there are dependencies here from *d7* (parsing the file system) or *d9* (performing file carving). Also if data came from a logical container or mobile phone extraction there may be dependencies there (*d13*, *d15*). If any of this is not done correctly then it could introduce error to this stage i.e. an incorrect hash. For example if:

- carving extracted only part of a file due to a footer occurring within the file contents.
- a carved file was originally fragmented with even just a single block in the middle belonging to a different file.
- the file system processing extracted incorrect data for a file, e.g., failing to take into account fix up arrays in NTFS resident data attributes [37].
- the file system processing extracted too much data for a file e.g., including file slack as part of a file.
- the file system processing did not recover enough data e.g., failed to follow FAT chains correctly.

Potential Error Introduced at this Stage:

- Computing the incorrect hash: the error actually introduced at this stage would be an implementation error in the hashing algorithm (INAC-AS/ALT).

4.9. Hash matching

Description: Hashed files are then often compared with entries in a predefined known hash database, either known to be associated with software, such as NSRL, or known bad, such as malware or known child abuse media.

Dependencies: The dependency here is on the file hashing process (*d17*) and, of course, everything before that.

Potential Error Introduced at this Stage:

- Failing to match a file because of a comparison error (INCOMP, INAC-AS).
- Failing to match a file because of an incomplete or inaccurate hashset (INCOMP).
- Incorrectly matching a file as innocuous because of an inaccurate hashset (INAC-AS).
- Error could also be introduced from a design perspective, if an insecure hashing algorithm is used (INAC-AS/ALT).

4.10. File Specific Processing

Description: File formats used as part of operating systems and applications contain open and proprietary data structures and embedded metadata, including: Registry hives, Event logs, SQLite and WAL files, email databases, compressed archives, encrypted containers, PDF and DOCX documents, scanned text (e.g., TIFF), binary plists, and EXIF headers, and many more. Each file format requires specialized processing to parse data structures and extract metadata.

Dependencies: There is a dependency here on identifying file type in order to trigger file specific processing (*d19*).

Potential Error Introduced at this Stage: Since there are such a broad range of processes included in this stage, specific errors are hard to include in this paper; SQLite alone could be discussed at length. Due to space constraints only a selection of examples are provided, illustrating each of the the generalized error types:

- Failing to recover SQLite records from the non-allocated space within a database, or failing to render aspects of a particularly obscure PDF component (INCOMP).
- Information being misunderstood or misrepresented, e.g., presenting event log data with the local time zone settings of the investigator's machine, rather than UTC or time zone of the system being examined (INAC-ALT), or more specifically, failing to present the translations that are being performed (MISINT).
- Extracting an incorrect target path from Windows Shortcut files (INAC-ALT/AS).

4.11. Keyword Indexing & Searching

Description: This is the process of attempting to identify relevant content using keywords. This is shown in the diagram in two stages: keyword indexing, and keyword searching. This may not always be the case and some tools (e.g., X-Ways) do allow keyword searching directly on the data without first building an index, but due to increasing data storage and complexity, most tools provide an indexing capability that first processes all files and extracted data and adds text to an index that enables near instant search results to be obtained.

Dependencies: In order for comprehensive text indexing to occur, since various file formats do not necessarily have directly accessible content (e.g., due to compression or formatting tags), files need to processed before data can be accessed for full indexing. This creates dependency *d24*. Some tools may also directly index text from unpartitioned or unallocated space, but this is not shown in the diagram for clarity.

Potential Error Introduced at this Stage: Within the keyword indexing process, since we assume content has been made accessible at the 'file specific processing' stage, the error here is associated only with building the index from the available text. There are, however, many considerations that affect whether text would be indexed, for example, use of text encodings, whether you facilitate substring searching, what you consider as word separators, how case sensitivity is handled, etc. [38], which could all result in data not being indexed (INCOMP). Another error that could be introduced would be incorrect mapping to source location (INAC-AS). Within the keyword searching process, error would be associated with retrieval of results from the index or parsing a supplied regular expression (INCOMP).

4.12. Timeline Generation

Description: Basic timeline generation involves extracting timestamps from the file system, but more comprehensive methods (e.g., Plaso) would also apply file specific processing and extract timestamps from files such as the Windows Registry, log files, SQLite databases that contain timestamps, etc.

Dependencies: For the file times, correct processing of the file system is required (*d16*), but all the other timestamps require the files to be correctly processed and timestamps extracted from within them (*d20*).

Potential Error Introduced at this Stage: For initial timeline generation the most likely errors are:

- Failing to extract timestamps from a file type that contains extractable low-level events (INCOMP)
- Incorrectly processing a file such that incorrect timestamps are extracted (INAC-ALT).

In addition, usually normalization is applied to the timestamps, which can introduce further sources of error:

- Incorrect timestamp conversion (either incorrect interval, epoch, or incorrectly parsing text strings containing timestamps) (INAC-ALT).
- Incorrectly applying a time zone (INAC-AS/ALT).
- Applying an interpretation to a timestamp that is incorrect (INAC-ALT/AS/), or overly simplified e.g., timestamp resolution (MISINT).
- Failing to consider an inaccurate clock in the timestamp's generation (INAC) or report its possibility (MISINT).

4.13. Timeline Analysis

Description: With a set of timestamps extracted, timeline analysis involves applying some interpretation to those timestamps [39]. For example, a USB connection causes multiple low-level events on a system, which during event reconstruction all contribute some aspect to the high-level event reconstruction of 'USB stick connected' [40].

Dependencies: For this analysis to be performed correctly, the correct timestamps need to have been extracted (dependency *d21*).

Potential Error Introduced at this Stage: Since this involves applying a set of rules or other interpretation process to the extracted timestamps, if the rules are not correct then this could result in an incorrect interpretation. Therefore potential error introduced at this stage are:

- Missing events that should be identified e.g., a rule set is excessively restrictive for an event that should be reconstructed, or incorrect (INCOMP)
- Identifying an event that did not occur e.g., a rule is incorrect and matches excessively (INAC-EX)
- Identifying an event at the wrong time (INAC-AS) [3]
- Assigning incorrect ancillary details to an interpreted event e.g., a USB stick was connected but presenting the serial number incorrectly (INAC-AS).

4.14. Geolocation Extraction and Analysis

Description: This type of analysis involves the correlation and, typically, visualization of data points that contain some geographic data, e.g., latitude/longitude or a postal addresses. This is shown in the diagram in two stages: geolocation extraction and geolocation analysis.

[3] Technically, this is a combination of missing an event and identifying one that did not occur, but presented separately for simplicity.

Dependencies: Extraction of location data can be performed either from the entire disk image without file system context, which is not shown in the diagram, or after extraction of data from specific files, e.g., EXIF extraction, SQLite database processing, or location information taken from some other log file, which all came from the 'file specific processing stage'. This is shown as dependency *d22*. The separate 'location extraction' stage shown would correlate and normalize all the location information. If those operations were not carried out correctly or data were missed, then that error propagates into the Geolocation Analysis.

Potential Error Introduced at this Stage: Some example errors that could be introduced at this stage:

- Rendering the location incorrectly on a map (INAC-AS).
- Interpreting a location stored for other reasons as one that was physically visited (INAC-AS, MISINT).
- Failing to express uncertainty of precise location in a visualization (MISINT).
- Failure in location lookup: e.g., an IP address being assigned a different location at the time of the investigation versus time of recording in the data (INAC-AS, MISINT).

4.15. Other Entity Extraction

Description: This process describes the extraction of other entities that do not fit into the categories of locations or timestamps. This could include details such credit card numbers or phone numbers [41]. In some tools e.g., Autopsy, this is expressed just as a specific keyword search using regular expressions, but is separated here for ease of discussion.

Dependencies: Due to the similarity with keyword searching, the dependencies are similar. The need to examine file content, which may be encoded in specific ways, makes this dependent on the file specific processing (*d26*), although entities could be extracted directly from the keyword index. It is also possible to apply entity extraction over the raw data from the disk image, but that is not shown in the diagram partly for readability, but also because such an approach is likely to return a limited subset of results without file specific processing.

Potential Error Introduced at this Stage: Within entity extraction all types of error can occur:

- Missing an entity that should be identified (INCOMP).
- Classifying an entity as another entity class (INAC-AS).
- Identifying an entity that was not present (INAC-EX).
- Misattributing an extracted entity to the wrong file or artifact (INAC-AS)

4.16. Automated Result Interpretation

Description: In order to make important information readily accessible to investigators, many tools apply automated processing to files or other artifacts, often databases, and present extracted results within the tool. This includes data such as 'messages sent', 'Google searches conducted', or 'contacts'.

Dependencies: In order to perform this process, the files containing the information must be made accessible (either processing the file system, or content carving), and processed according to their type (for example non-allocated records may be

recovered from SQLite databases) under 'file specific processing', which creates dependency *d28*.

Potential Error Introduced at this Stage: Since this is the application of rules to files or artifacts to extract content such as 'calls made', 'contacts' or 'web search', any incorrect rules coded in the tool could introduce error, as does incorrect or out of date knowledge on which the rules are based (INAC-EX/AS, MISINT). Therefore, failing to identify and consider the version of the application being automatically parsed could miss entries (INCOMP), create spurious ones (INAC-EX), or extract entries with incorrect details (INAC-ALT).

4.17. Image Processing (images and videos)

Description: This stage summarizes the use of any image processing techniques to images or video. Some examples include: Skin tone detection in X-Ways Forensics, Find similar pictures in AXIOM, Video Triage plugin for Autopsy, or AI object recognition.

Dependencies: Since these processes need to be applied to images and videos, correct identification of those types of file leads to dependency *d31*. However, there may also be images or videos embedded in other formats, and, thus, the 'file specific processing' stage, where an image may be extracted from a zip file or Office document, is also a dependency (*d30*).

Potential Error Introduced at this Stage: Error at this stage depends on the specific image processing that occurs. However, most apply some sort of categorisation or identification, and should have some level of confidence or uncertainty attached to the output. Failure to express this could result in error (MISINT). Error can also occur if thresholding is too lenient (INAC-EX/AS) or too strict (INCOMP). Failure to take into account image rotation or other manipulations may also result in missed images (INCOMP).

4.18. Mismatched Signature Detection

Description: This stage allows detection of files that have an extension that does not match the actual content of the file.

Dependencies: This relies on the file type identification to have occurred (dependency *d18*) so this can be compared with the file extension extracted earlier on in the processing flow.

Potential Error Introduced at this Stage: Error here would be misclassifying a file as having an incorrect extension when it does not (INAC-AS), or missing a file that has an incorrect extension (INCOMP). This could be the result of an implementation error, but also from inaccuracy in the database on which the file type identification was based.

4.19. Content Flagging Using AI

Description: This newer type of processing is usually to assist investigators locate text content (at present) of a particular kind within a large set of messages, e.g., grooming language. However, this is a fast-moving area and needs substantial more discussion than space allows.

Dependencies: Dependency *d29* shows that for text chat to be processed by an AI model the text must first be extracted, which requires file specific processing, mostly likely of a chat message database.

Potential Error Introduced at this Stage: This is similar to that described in the image processing section where any flagged content should have some level of confidence or uncertainty attached to the output. Failure to express this could result in error (MISINT), and again, error can also occur if thresholding is too lenient, resulting in content being flagged incorrectly as suspicious (INAC-EX/AS) or too strict and missing content (INCOMP).

5. Recommendations

From the previous section it is now possible to clearly see that there are multiple distinct stages to processing a digital evidence source. It is also possible to see that there are dependencies between these stages, and specific errors associated with each of those processes.

As a result of this analysis, in order to improve the digital forensic process, it is proposed that it is possible within a tool to output the results of each stage of processing in a standard representation. The deconstruction above in Section 4 provides a suggested starting point for where output would be valuable if was implemented. Due to space constraints the specific output cannot be discussed in detail in this paper, but Table 2 in the appendix contains starting suggestions for what each stage of processing could output. This approach would provide benefits in the following areas within digital forensics:

Improvements to tool testing and validation process: If tools were able to output the results of each of these layers of abstraction then it would be easier to construct tests on particular aspects of tool use. At the moment, testing higher levels e.g., keyword searching, is difficult because lower levels could cause differences in results. If the lower levels were exported and were in a standardized format then the ground truth data at each stage could be compared and the source of any discrepancy much more easily identified. For example, it may be that an automatically extracted contact is missed because of a problem with the automated result interpretation part of the process (a database query error or limitation for example), or it may be that a non-allocated file that contains that information was not recovered correctly earlier on in the process, which would be at dependency point *d6*, or that a file type was not identified correctly, at dependency *d19*.

Also if we wanted to answer questions of reliability of keyword search results e.g., are they 'complete', the factors affecting that completeness are now more easily visible, e.g., are all the files extracted? Are file types correctly identified? Is correct file specific processing applied for all files e.g., pdf/docx? Only then can we start to think about if the particular subtleties of the keyword indexing are correctly implemented e.g., character encoding, storage, text indexing, and keyword matching within that index. Representing the results of each stage of processing in a standard form would also facilitate automated tool testing.

Error-focused test datasets: Linked to validation and tool testing, increasing visibility and understanding of errors at each

abstraction layer can assist in creating test datasets that focus on specific errors at each abstraction layer that are otherwise hidden within digital forensic tools and are less likely to be exposed by more general test datasets. This is more efficient since creating many small datasets that each target specific errors is modular and scalable, and is extensible to encompass future features that implement AI-based methods. This would also help developers identify which abstraction layer has a problem rather than simply knowing that a monolithic tool is not producing the correct result, and therefore find and fix the problem. Error-focused test datasets can be made modular to support tool validation testing as part of the continuous integration and continuous delivery process.

Dual tool verification during a case: Similar to the above examples, if a dual tool approach is used during an investigation, it can be challenging to identify and explain differences in tool output. Comparison of structured output at each of these layers would make it more feasible to determine why tools disagree, and which is the correct result, and could also be performed automatically. Also, if practitioners must use a tool that has an identified error, knowing the specific abstraction layer helps them avoid the error and develop work-around solutions.

Historic case re-evaluation: In laboratories using ISO17025 or similar quality control processes, and as documented in [42], it should be documented exactly what version of a tool was used in a case. If a problem is subsequently identified in an older tool version, and it is possible to identify which subprocess the issue affects, potential issues can be tracked not just to the particular cases that used that version of the tool, but it would be possible to understand which results that error affects, rather than the whole case.

Improved artifact provenance: Given the large number of processes that the raw digital evidence goes through before a useful digital forensic artifact is accessible, this shows the importance, and provides a basis for tracking the full provenance of a result through all these abstraction layers, along with their error, back to the raw data from which they were derived [43] [39]. Provenance is an integral part of CASE, using the `InvestigativeAction` and `ProvenanceRecord` representations to keep track of when, where and who used which tools to perform investigative actions on data sources, and the result. When a tool is used, the version and configuration parameters can be included in provenance documentation as demonstrated in the CASE example for this paper. Furthermore, CASE has been projected into the PROV-O space[4], re-interpreting CASE concepts through a semantic framework focused on provenance. The CASE–PROV-O implementation[44] enables automatic consistency checking of CASE data with unit tests to ensure that it conforms to PROV-CONSTRAINTS [45], including detecting broken chain of evidence.

Education: With these distinct processes clearly identified, discussing with students the nature of those processes, their limitations and challenges should be easier to structure and to set in the context of the broader overall investigation.

Research: With these processes isolated, research areas for tool capability improvements are more clearly defined, and it should also be clearer which other processes supply data to a specific process of interest, and what processes may depend on the output. This latter part is particularly important when new processes developed have some level of uncertainty, e.g., non-allocated file recovery, or file fragment identification and reassembly, as that uncertainty should to be conveyed to any subsequent processes that may depend on that data.

6. Demonstration

To demonstrate the proposed approach and its value, an error-focused test dataset was fabricated with three active partitions and a fourth partition that was deleted to leave a "lost" partition containing a file named "missedme.txt" and a non-allocated file named "first.txt" that was overwritten by a file named "second.txt" (dataset available online[5]). A CASE representation of the ground truth dataset was constructed and is available here[6]. Table 1 shows the results for two forensic and one data recovery tool examining the same disk image. This was created from manual review of the tool results, checked against the ground truth. The tools mentioned are real tools, but are presented here anonymously as Tool 1 at version 1.2.3, Tool 2 at version 2.4.6, Tool 3 at version 3.5.8, and (in the appendix) Tool 4 at version 13.0.

Considering the second part of the table only, i.e. a high-level test for correct file recovery, it shows that Tool 1 attributes all recovered files correctly and highlights the ambiguity of file content for first.txt; Tool 2 does not locate the files at all; and the data recovery tool Tool 3 identifies the files, but incorrectly associates content from second.txt to first.txt. From these file results alone it is not possible to understand the nature of the error in Tool 2 or Tool 3. However, if output was available in standard representation for the earlier dependency (identify partitions), and if we therefore consider the whole table, which includes that previous stage, problems can be seen earlier in the dependency chain, and the work needed to correct the error for Tool 2 is actually in partition recovery rather than file recovery.

Manual representations of the results for each of these tools are provided in the repository, along with an automated comparison with the ground truth, which shows how comparison against ground truth and other tools would be more efficient if this output was exported programmatically by tools in a standard format. This demonstrates that with this abstraction layer model, and standardised output at each stage of processing, tool differences are more easily identifiable, and errors when compared with documented ground truth are made transparent.

7. Limitations and Further Work

In terms of limitations, an 'outside in' approach was used to look at forensic tools, no reverse engineering of forensic tools

[4] https://www.w3.org/TR/prov-o/

[5] https://github.com/chrishargreaves/digital-forensic-tool-abstractions

[6] See https://github.com/casework/CASE-Examples/tree/master/examples/illustrations/partitions#partitions-examples

Ground Truth Tests	Tool 2	Tool 3	Tool 1
IDENTIFY PARTITIONS			
P1 FAT32 identified	y	y	y
P1 start/end ok	y	y	y
P1 status=live	y	y	y
...			
P4 FAT32 identified	INCOMP	y	y
P4 start/end ok	INCOMP	y	y
P4 status=del	INCOMP	y	y
IDENTIFY FILE SYSTEM AND PROCESS FILES			
P4/missedme.txt exists	INCOMP	y	y
P4/missedme.txt content ok	INCOMP	y	y
P4/first.txt exists	INCOMP	y	y
P4/first.txt content flagged NA	INCOMP	INAC-AS	y
P4/first.txt uncertainty presented	INCOMP	MISINT	y
P4/second.txt exists	INCOMP	y	y
P4/second.txt content ok	INCOMP	y	y

Table 1: A summary of tool results for processing a test dataset. Illustrates that output at different stages of processing helps identify the source of error.

was performed, nor were forensic tool providers interviewed. It is likely there are other 'digital forensic analysis tool components' that exist, however they can be simply added to the existing dependency diagram. It is also likely that some tool vendors do not have the exact same extrapolated dependencies, although most can be logically argued that they must be performed for subsequent stages to take place. If there are significant differences, we would encourage tool providers to provide a similar specific diagram indicating the precise data flow between high level components of their software.

The process of modeling inherently has the limitation of simplification. Like digital forensic process models, many of which present a slightly different perspective or level of granularity, this model too could be extended, have more detail added, breaking down one process into many. Processing of specific file types could also have internal process dependencies extrapolated. However, for the purposes of this paper, the appropriate level of abstraction and granularity was chosen to introduce this concept, provide sufficient detail for discussion, show how errors can propagate, and present an option for vendors to help monitor these potential errors. Future work could certainly expand some of these stages, or add in additional processes based on tools that were not reviewed as part of this research.

Another limitation is that it is unlikely that this paper has captured all the errors that are possible for each of the processes discussed. A comprehensive discussion of all errors cannot be guaranteed, however, as there was no documentation found that examines the overall tool process even at the level of abstraction achieved in this paper, it is considered helpful to increase understanding of error propagation in digital forensic tools.

While examples of the output that would be useful to export at each stage are provided in the appendix, it has not been formally defined in an interchange format such as CASE. Not all of the points discussed are currently modeled in CASE and due to space limitations this was not possible to cover in this paper.

While this paper has deliberately focused on exposing and mitigating error within the extraction and analysis performed by digital forensic tools, a broader examination of error across all stages of the digital forensic process could also be conducted. The other future work that this facilitates, is that while in this paper the dependencies were used to highlight propagation of error, this may also provide a suitable basis for consideration of the propagation of uncertainty. Reference [10] discusses that often error in digital investigations "cannot be expressed as a definite value, e.g., x ± y, but can be expressed as uncertainty (possible error) in inferred events, i.e. alternative possible hypothesised events that explain the current state of the examined digital evidence." When a tool reports, for example, that a specific file was recovered, this is actually a hypothesis that some tools test more than others. In [32] a standard approach is provided for hypothesis testing and treating uncertainty in the results of file recovery operations. This treatment of uncertainty can be extended to tool results in general, including AI-based results [46]. When a result from a tool is presented, and we consider the hypothesis of 'the tool result is correct', if all the abstraction layers through which the data has passed are defined, and all the possible errors that could occur at each of those layers are listed, the compiled list of all those potential errors are actually a list of alternative hypotheses to 'the tool result is correct'. This is a starting point for systematic reasoning about uncertainty in tool results, and a foundation for improved quality in digital forensic science.

8. Conclusions

Since digital forensic tools are necessary given the volume and complexity of digital evidence in modern digital investigations, it is important that results produced are accurate, complete, and do not cause misinterpretation. It is already challenging within tools to constantly and consistently differentiate fact from inference and to identify potential error and uncertainty in tool results. Tool vendors have an opportunity to do more than extract and present results; they could document abstraction layers and output intermediate data that follow ASTM E3016-18 guidance for error mitigation analysis, and make errors within tools more obvious to examiners so that misinterpretations are minimized. This paper has shown that to achieve this, the full dependencies and history of abstraction layer translations need to be exposed. To improve transparency and facilitate comparison, it has also argued for the potential value of exporting the output of the abstraction layers as digital evidence moves through the internal processes of digital forensic tools. This paper has, therefore, provided a contribution towards further understanding error in digital forensic tools, identifying it when it occurs, and mitigating it.

References

1. Garfinkel SL. Digital forensics research: The next 10 years. *digital investigation* 2010;7:S64–S73.
2. Quick D, Choo KKR. Impacts of increasing volume of digital forensic data: A survey and future research challenges. *Digital Investigation* 2014;11(4):273–294.

3. Test Environment and Procedures for Testing SafeBack 2.18. Tech. Rep.; NIST; 2003. URL: https://www.nist.gov/system/files/documents/2017/05/09/SafeBack-2-18-Procedures.pdf.
4. Du X, Hargreaves C, Sheppard J, Anda F, Sayakkara A, Le-Khac NA, Scanlon M. Sok: Exploring the state of the art and the future potential of artificial intelligence in digital forensic investigation. In: *Proceedings of the 15th International Conference on Availability, Reliability and Security*. 2020:1–10.
5. Beckett J, Slay J. Digital forensics: Validation and verification in a dynamic work environment. In: *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)*. IEEE; 2007:266a–266a.
6. UK Forensic Science Regulator . Forensic science regulator guidance: Method validation in digital forensics, fsr-g-2018 issue 2. 2020. URL: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/921392/218_Method_Validation_in_Digital_Forensics_Issue_2_New_Base_Final.pdf; accessed: 2023-09-10.
7. Casey E. Error, uncertainty and loss in digital evidence. *International Journal of Digital Evidence* 2002;1(2).
8. Carrier B, et al. Defining digital forensic examination and analysis tools using abstraction layers. *International Journal of digital evidence* 2003;1(4):1–12.
9. Lyle JR. If error rate is such a simple concept, why don't i have one for my forensic tool yet? *digital investigation* 2010;7:S135–S139.
10. Hargreaves C. Assessing the reliability of digital evidence from live investigations involving encryption. Ph.D. thesis; Cranfield University, UK; 2009.
11. Gruber J, Hargreaves CJ, Freiling FC. Contamination of digital evidence: Understanding an underexposed risk. *Forensic Science International: Digital Investigation* 2023;44:301501.
12. ASTM . Astm e3016-18 standard guide for establishing confidence in digital and multimedia evidence forensic results by error mitigation analysis. 2018. URL: https://www.astm.org/e3016-18.html.
13. Garfinkel S. Lessons learned writing digital forensics tools and managing a 30tb digital evidence corpus. *Digital Investigation* 2012;9:S80–S89.
14. NIST . Computer forensic reference dataset portal (cfreds). undated. URL: https://cfreds.nist.gov.
15. Du X, Hargreaves C, Sheppard J, Scanlon M. Tracegen: User activity emulation for digital forensic test image generation. *Forensic Science International: Digital Investigation* 2021;38:301133.
16. Wilsdon T, Slay J. Validation of forensic computing software utilizing black box testing techniques. In: *4th Australian Digital Forensics Conference*. 2006:doi:10.4225/75/57b13e59c705b.
17. Michelet G, Breitinger F, Horsman G. Automation for digital forensics: Towards a definition for the community. *Forensic Science International* 2023;349:111769. URL: https://www.sciencedirect.com/science/article/pii/S0379073823002190. doi:https://doi.org/10.1016/j.forsciint.2023.111769.
18. Carrier, Brian . Digital forensics tool testing images. 2010. URL: https://dftt.sourceforge.net.
19. Duns H, Lawton D. Ground truth datasets for data recovery techniques in digital forensics, dstl/doc136843. 2022. URL: https://cfreds.nist.gov/all/HollyDuns%2FDSTL/GroundTruthDatasetsSupportingDocuments; accessed: 2023-09-10.
20. Horsman G. Tool testing and reliability issues in the field of digital forensics. *Digital Investigation* 2019;28:163–175.
21. Horsman G. "i couldn't find it your honour, it mustn't be there!"–tool errors, tool limitations and user error in digital forensics. *Science & Justice* 2018;58(6):433–440.
22. Glisson WB, Storer T, Buchanan-Wollaston J. An empirical comparison of data recovered from mobile forensic toolkits. *Digital Investigation* 2013;10(1):44–55.
23. Freiling F, Glanzmann T, Reiser HP. Characterizing loss of digital evidence due to abstraction layers. *Digital Investigation* 2017;20:S107–S115.
24. Schneider J, Deifel HP, Milius S, Freiling F. Unifying metadata-based storage reconstruction and carving with layr. *Forensic Science International: Digital Investigation* 2020;33:301006.
25. Ellison D, Ikuesan RA, Venter HS. Ontology for reactive techniques in digital forensics. In: *2019 IEEE Conference on Application, Information and Network Security (AINS)*. IEEE; 2019:83–88.
26. Brady O. Exploiting digital evidence artefacts: finding and joining digital dots. Ph.D. thesis; King's College London; 2018.
27. NIST . Computer forensics tools & techniques catalog. ???? URL: https://toolcatalog.nist.gov/taxonomy/index.php; accessed: 2023-09-10.
28. Metz J. Ewf specification from libewf project. 2023. URL: https://github.com/libyal/libewf/blob/main/documentation/Expert%20Witness%20Compression%20Format%20(EWF).asciidoc#.
29. Cohen M, Garfinkel S, Schatz B. Extending the advanced forensic format to accommodate multiple data sources, logical evidence, arbitrary information and forensic workflow. *digital investigation* 2009;6:S57–S68.
30. Gupta MR, Hoeschele MD, Rogers MK. Hidden disk areas: Hpa and dco. *International Journal of Digital Evidence* 2006;5(1):1–8.
31. Turner P. The remapped/reallocated sector conundrum from a digital forensic investigation and data sanitisation perspective, presented at dfrws usa 2023. 2023. URL: https://dfrws.org/presentation/the-remapped-reallocated-sector-conundrum-why-are-remapped-sectors-important/.
32. Casey E, Nelson A, Hyde J. Standardization of file recovery classification and authentication. *Digital Investigation* 2019;31:100873.
33. Pal A, Memon N. The evolution of file carving. *IEEE signal processing magazine* 2009;26(2):59–71.
34. Casey E, Zoun R. Design tradeoffs for developing fragmented video carving tools. *Digital Investigation* 2014;11:S30–S39.
35. Dubettier A, Gernot T, Giguet E, Rosenberger C. File type identification tools for digital investigations. *Forensic Science International: Digital Investigation* 2023;46:301574.
36. Magazinius J, Rios BK, Sabelfeld A. Polyglots: crossing origins by crossing formats. In: *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. 2013:753–764. doi:10.1145/2508859.2516685.
37. Carrier B. File system forensic analysis. Addison-Wesley Professional; 2005.
38. Guo Y, Slay J, Beckett J. Validation and verification of computer forensic software tools—searching function. *digital investigation* 2009;6:S12–S22.
39. Hargreaves C, Patterson J. An automated timeline reconstruction approach for digital forensic investigations. *Digital Investigation* 2012;9:S69–S79.
40. Patterson J, Hargreaves C. The Potential for cross-drive analysis using automated digital forensic timelines. In: *6th Cybercrime Forensics Education and Training*. 2012:.
41. Garfinkel SL. Digital media triage with bulk data analysis and bulk_extractor. *Computers & Security* 2013;32:56–72.
42. Hildebrandt M, Kiltz S, Dittmann J. A common scheme for evaluation of forensic software. In: *2011 Sixth International Conference on IT Security Incident Management and IT Forensics*. 2011:92–106. doi:10.1109/IMF.2011.11.
43. Casey E, Barnum S, Griffith R, Snyder J, van Beek H, Nelson A. Advancing coordinated cyber-investigations and tool interoperability using a community developed specification language. *Digital investigation* 2017;22:14–45.
44. CASE Ontology . Case-implementation-prov-o. 2023. URL: https://github.com/casework/CASE-Implementation-PROV-O.
45. Nies TD, Cheney J, Missier P, Moreau L. Constraints of the prov data model. Recommendation; W3C; 2013. URL: https://www.w3.org/TR/prov-constraints/.
46. Casey E, Bollé T. Formalising representation and interpretation of digital evidence to reinforce reasoning and automated analysis. *Artificial Intelligence (AI) in Forensic Sciences* 2023;:74.

9. Appendix

9.1. Tool Testing Details

This section provides the specific results from testing tools against the error-focused dataset in Section 6. Results are summarized in Table 3.

Using this error-focused test dataset, Tool 1 lists three active partitions, has a feature that finds the lost partition. It, therefore, finds the files within and classifies “first.txt” as “prev. existing, 1st cluster not available” and shows that the cluster is allocated to “second.txt”. It also displays a specific popup message that warns file contents the tool ascribes are not necessarily the original file contents. Tool 1 does not automatically display the extended partition in the MBR partition table, a sign that another partition previously existed. Tool 2 lists the three active partitions but does not automatically recover the lost partition and, therefore, misses the files it contains. Another tool can be used to extract the lost partition, which can be loaded into Tool 2 for examination; but Tool 2 does not correctly classify “first.txt” as overwritten, causing potential MISINT because the displayed content is actually from “second.txt”.

In Table 3, “Type” is omitted for the Disk Management step because Disk Management’s interface makes no distinction between the partition’s type code and the requested file system. Index 4 under Disk Management is the Extended Partition Table partition, which Tool 4 also presents; but Tool 1 does not present this, instead presenting the recovered partition containing the FAT file system.

Stage	Suggested Output
Parse Image Format	Hash(es) of the raw data contained within the image.
Validate Disk Image	The hash of the data compared and a log of what they were compared with to perform the validation (internal hash, externally supplied information
Identify Partitions	List of partitions with start and end sectors, including any deleted partitions recovered.
Process File System	List of files recovered from a file system, including any non-allocated but recoverable files, capturing any uncertainty associated with the recovery.
Identify Content (Carving)	List of any files recovered, their locations on the disk or within a file, the process used for identification/reassembly, e.g., headers/footers.
File Type Identification	Type associated with each file and the means by which that was derived.
File-Specific Processing	Dependent on the specific file processing, and one of the most challenging to determine what to export and how to represent it; but, as examples: for SQLite deleted record recovery, the offset of the identified record and cells and cell types identified; for documents, a listing of the objects (text, images, etc.) contained within; or for EXIF data, the extracted fields.
File Hashing	A list of all files and their hashes.
Hash Matching	A list of all files matched to a hash set, the hashes, and the origin of the hashset used.
Mismatched Signature Detection	The file, extension, file signature and reason for mismatch.
Timestamp Extraction	List of all timestamps extracted, how they were extracted (file time, SQLite database row, offset within a file, log entry number, etc.), and any time offsets applied.
Timeline Analysis	A list of interpreted events, the low-level events on which they were based, or references to them, and the rules or processes used to infer this event.
Geolocation Extraction	List of all locations extracted, how they were extracted (e.g., SQLite database row, offset within a file), any potential uncertainty associated with the location.
Geolocation Analysis	The results of any analysis applied to the raw locations, along with any algorithms used to infer movement, etc., and their uncertainty.
Keyword Indexing	Exporting a representation of all the keywords and where they were identified would be substantial. However, at a more basic level, the parameters used to generate the index, e.g., character encodings used or not used, and other indexing settings could be expressed.
Keyword Searching	The keyword that was used for the search and the corresponding results.
Other Entity Extraction	The extracted entity, its provenance, and method used to identify it.
File Browsing and Filtering	If a filtered listing is exported, the details of the filter should be exported along with the listing.
Automated Result Interpretation	The automated result, the process used for extraction, e.g., the source file and the SQLite query used for extraction.
Image Processing	Depends on the specifics, but, for example, the bounding box of an identified face within an image.
AI-Based Content Flagging	The identified content along with any assigned labels from the AI process, along with confidences.

Table 2: Suggested Output for Each Stage of Processing. These are starting suggestions for useful output at each stage of processing, but are not exhaustive.

	Disk Management			Tool 1			Tool 4		
Index	UUID	Type	Offset	UUID	Type	Offset	UUID	Type	Offset
1	1ef219fe-...			d34878c1-...	0c	65536	18ddcc3a-...	0c	65536
2	251032a2-...			7868ea24-...	07	104923136	27073ab0-...	07	104923136
3	32f65e72-...			7aad22df-...	07	230752256	374fdbc6-...	07	230752256
4	4f68c4c7-...			5715fe11-...			e8be10dc-...	05	377552896
5	5000a2e8-...								

Table 3: A comparison of partitions known to be generated for an experimental USB storage device, with the last partition deleted before processing the image with forensic tools. “Disk Management” represents the data generation process using the built-in Windows Graphical User Interface (GUI) tool and, thus, the list an ideal forensic tool would recover. “Index” is the position number in the list of partitions recovered by the forensic process (or generated by Disk Management). “UUID” is the beginning of the graph-individual’s universally unique identifier (UUID). “Type” is the partition type code drawn from the partition table. Offset is in bytes.