practices, it is necessary to give due consideration to alternative claims, and specifically search for contradictory facts (falsification). In the above case example, it is necessary to evaluate the opposing claim "the non-allocated photograph was not fully recovered."

Even when digital traces lead to seemingly incontrovertible conclusions, it is important for forensic practitioners to keep in mind that subjectivity and some level of uncertainty are involved in the evaluation of forensic findings. When forensic practitioners concentrate on proving or disproving a specific claim, there can be a risk of confirmatory bias. To mitigate this risk, an increasing number of best practice guidelines are instructing forensic practitioners to evaluate the strength of evidence given one claim versus a given alternative claim. Continuing the case example above, using language in the ENFSI Guideline for Evaluative Reporting in Forensic Science [12], a forensic practitioner may state: "In my opinion, the results observed in my forensic examination of the file system data structures and recovered content are exceedingly more probable if the non-allocated photograph was Fully Recovered, than if the recovered content was from an unknown file." Statements in this form clearly convey that forensic practitioners have made an evaluation of digital traces in their domain of expertise, and express the relative strength of evidence given one claim versus the opposing claim. As with any expert opinion, such statements require an explanation of the forensic examination of digital traces, and any contextual information that was taken into consideration. Such transparency is necessary to assess expert opinions, particularly when forensic practitioners have differing levels of confidence in forensic results.

The Bayesian approach has gained traction in certain forensic disciplines, producing a ratio of the probability of evidence given alternative opposing claims, called a likelihood ratio (LR). However, further work is needed to apply this approach to digital forensics.

5. Classification and Authentication of File Recovery

A taxonomy for classifying the recoverability of files is provided in Table 1. This taxonomy defines terminology for distinct types of file recovery. For each class in Table 1, an example of common situations for content estimation is summarized, and more detailed discussion of each class is provided in subsequent sections. The classes with recovered, or potentially recovered, content have the implicit requirement that there is no indication that the

associated clusters have been overwritten by a newer file. In addition, all files with recovered content can have a sub-classification of Fragmented when contents are stored in non-contiguous storage regions which can make recovery more difficult and error prone. A sub-classification of Encrypted could also be applied to files with recovered content that is inaccessible due to encryption. All classes with recovered file system metadata could also have a sub-classification of Name Partially Overwritten when the first character of the name is overwritten such as on FAT file systems. All of these classes of file recovery can have a sub-classification of Path Not Recovered orPath Not Fully Recovered when the file path (parent folder) is unknown, or is only partly recovered.

Table 1: Taxonomy for classification of file recovery results with the associated status of name, metadata, and content

Class	**Name Status**	**Metadata Status**	**Content Status**	**Example File trace**
Fully Recovered	recovered	recovered	recovered	Resident MFT
Potentially Recovered	recovered	recovered	potentially recovered	Non-resident MFT
Partially Recovered	recovered	recovered	unknown/ overwritten	File system metadata and the first part of the content recovered, but other fragments are unknown or overwritten
Name and Content Recovered	recovered	unknown/ overwritten	recovered	Only recovered Name and Content characteristics are recovered (e.g., Alternate Data Stream), and file system metadata are not recovered
Metadata and Content Recovered	unknown/ overwritten	recovered	recovered	Only the Name of file is not recovered, but recovered metadata and content characteristics are compatible
Content Recovered	unknown/ overwritten	unknown/ overwritten	recovered	Only content recovered using carving techniques
Metadata and Name Recovered	recovered	recovered	unknown/ overwritten	Only Metadata and Name recovered from INDX entry, `$LogFile` record, Windows LNK file, Registry
Metadata Recovered	overwritten	recovered	unknown/ overwritten	Only the inode metadata of file recovered
Name Recovered	recovered	unknown/ overwritten	unknown/ overwritten	Only the name of file recovered in an EXT directory structure

Currently, digital forensic tools do not have a generally agreed upon way to represent the results of file recovery operations. To show the different ways

Table 2: Comparison of Multiple Tools Icons and Annotation of File Recovery Results. An x indicates that an icon with a red "x" is displayed by the tool.

Class	Autopsy 3.1.1	AXIOM 2.4.0	EnCase 7	FTK 6	XWays Forensics 18.2
Fully Recovered	x	"Deleted"		x	[faded file icon] "previously ex. file, contents unchanged"
Potentially Recovered	x	"Deleted"		x	? "previously ex. file, contents may have changed"
Partially Recovered	x	"Deleted, Overwritten"	x	x	x "previously ex. file, 1st cluster not available"
Name and Content Recovered	x	"Deleted"		x	ADS treated as file
Metadata and Content Recovered	N/A	N/A	N/A	N/A	N/A
Content Recovered	(listed under `$CarvedFiles`)	"Carved" folder	"Carved" in name	"Carved" in name	C "file, carved from sectors"
Name and Metadata Recovered	x	"Deleted" (in Artifact View of `$LogFile`)	N/A	N/A	? "file contents unknown"
Metadata Recovered	x	"Deleted"		x	? "only metadata available"
Name Recovered	x	"Deleted"	N/A	N/A	x "previously ex. file, 1st cluster not available"

that digital forensic tools portray file recovery results, Table 2 summarizes the visual cues and annotations used by five tools.

Table 2 shows that some tools use the same visual cue for all classes of file recovery, regardless of their recoverability status. Other tools use a distinct cue to distinguish Partially Recovered files. X-Ways Forensics uses different descriptors for each class of file recovery in Table 2.

> *"Deleted files and directories are represented in the directory browser with lighter icons. Icons with a blue question mark indicate that the original file or directory content may be still available. Deleted objects that WinHex [also XWays Forensics] knows*

are no longer accessible (either because their first cluster has been reallocated, because it is unknown, or because they have a size of 0 bytes) have icons crossed out in red." [13]

The following sections provide examples of each class listed in Table 1 above, and discuss the authentication process for each type of file recovery.

5.1. Fully Recovered

A file is classified as Fully Recovered if a sufficient level of confidence is reached for the name, file system metadata and file content recovery. An example would be when the file system entry can be recovered and the original file content are unchanged, such as when the complete file content are resident in a recovered MFT `FILE` entry. In this example, the original data are unchanged and no content estimation is involved in the recovery. A sample standardized representation of this class of file is provided in Listing 2 of Appendix B, as discussed further in Section 9.

As another example, when a copy of a file is stored in a Volume Shadow Copy (VSC), or any form of file system snapshot that contains backup copies of files, the file can be classified as Fully Recovered. Such Fully Recovered files may be found on other forms of file system snapshots, including mobile backups and files that have been synchronized with other devices or cloud storage. An example is a digital photograph that no longer exists on the originating mobile device, but that can be recovered from a cloud backup of that device.

5.2. Potentially Recovered

A file is classified as Potentially Recovered if there is no indication that clusters previously allocated to the non-allocated file have been overwritten by a newer file, but cannot confirm that original file has not changed. A sample standardized representation of this class of file is provided in Listing 3 of Appendix B, as discussed further in Section 9.

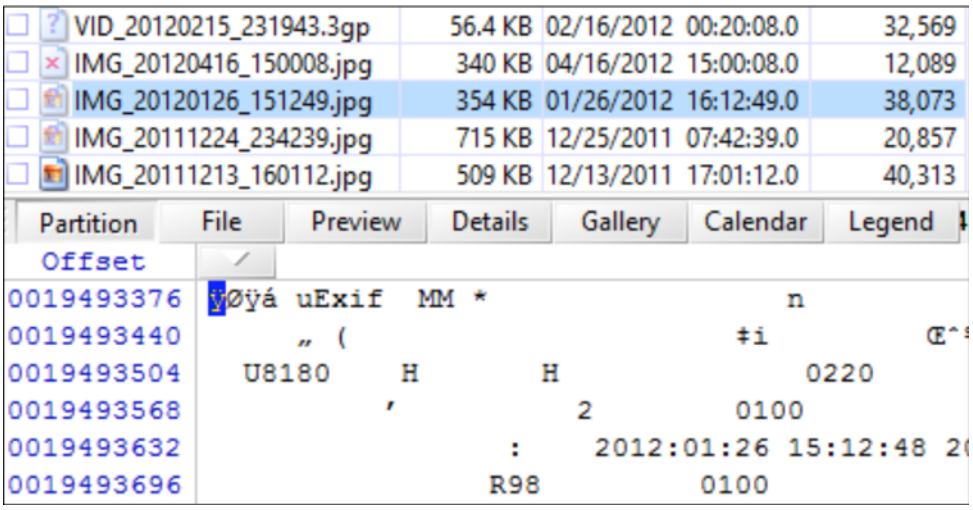

Figure 2: A deleted file that is Potentially Recovered, viewed using XWays Forensics

Scenario 1: Authentication of Potentially Recovered content leads to decision of Fully Recovered

In certain contexts, it might be possible to promote a non-resident file from Potentially Recovered to Fully Recovered, even if there is no way to definitively determine that the original file content is recovered. For instance, if the recovered content contains traits (e.g., header signature AND content size value in the header AND creation timestamp in the header) that are all compatible with the file system metadata, then a digital forensic tool or practitioner might have sufficient confidence to assert that the file content is recovered. As discussed in Section 4 above, the evaluation process underlying this promotion involves the subjective expert opinion of the forensic practitioner. Some forensic practitioners require stronger support in order to promote a file from Potentially Recovered to Fully Recovered, such as a Zip file that contains embedded CRC values that permit content verification.

For some Potentially Recovered files, there can be indications that the original file content is still present. For instance, the characteristics in the file content such as header and size might match the file system metadata, signifying that the original file content still exists. However, it is not safe to assume that a compatible header at the beginning of a non-allocated file

indicates that the file was not overwritten. There is a possibility that the file was overwritten with a new file of the same type, or that later portions of the non-allocated file were overwritten. For example, in Fig. 2, the file named `IMG_20120126_151249.JPG` is classified as Potentially Recovered. Further analysis of the Exif header information reveals characteristics that are compatible with the non-allocated file's name, creation date and size. Comparison of the picture content with embedded geolocation information could provide further indications that the recovered non-allocated file is, in fact, the original digital photograph. Additional information could provide even stronger support that the original content was recovered, such as comparing the full size picture to the associated embedded thumbnail and Windows thumbnail cache. In this situation, a forensic practitioner might evaluate the totality of available digital traces and assert that "In my opinion, the results observed in my forensic examination of the file system metadata, date in the file's name, creation dates in Exif header information, size recovered, content, embedded thumbnail, Windows thumbnail cache are far more probable if the non-allocated photograph was Fully Recovered, than if the recovered content was from an unknown file."

There may be other ways to raise the level of confidence that a Potentially Recovered file is actually Fully Recovered, including file content validation (file opens and renders properly) and comparison with a file of known SHA256 hash value.

When authenticating Potentially Recovered files, care must be taken to ascertain whether any part of the original content of the file was overwritten.

Scenario 2: Authentication of Potentially Recovered content leads to decision of Partially Recovered

When forensic examination of file recovery results reveals that recovered content is not compatible with the recovered file name or metadata, it may be necessary to demote a file from Potentially Recovered to Partially Recovered. For instance, if recovered file system metadata indicates that the file is a JPEG but the recovered content contains some other type of data (e.g., HTML), then a forensic practitioner or tool might have sufficient confidence to assert that the original file content was not recovered thereby reclassifying the file recovery results as Partially Recovered. The associated assertion might be "In my opinion, the results observed in my forensic examination of the file system data structures and recovered content are exceedingly more probable if the non-allocated photograph was Partially Recovered, than if the recovered content was from the original file."

Authentication of files initially ascribed to other classes can result in their promotion to Potentially Recovered as outlined in in Section 6 below.

Arguably, files that are initially classified as Potentially Recovered only exist pre-authentication, in a form of file system purgatory because authentication of such files will result in their ascription to one of the other classes in Table 1.

5.3. Partially Recovered

A file is classified as Partially Recovered if available file system metadata indicates that a newer file has been allocated to some clusters that were previously allocated to the non-allocated file, but remaining file fragments can be recovered.

In the simplest case, when clusters that were allocated to the non-allocated file are reallocated to a newer file, most digital forensic tools mark the non-allocated file with a distinctive icon indicating that it has been overwritten, and display the complete path of the new file. A sample standardized representation of this class of file is provided in Listing 4 of Appendix B, as discussed further in Section 9.

Methods for determining whether any part of a file has been overwritten depend on the type of file system. For instance, NTFS uses the `$BitMap` file to keep track of which clusters are allocated, but FAT does not have

such a tracking mechanism. Therefore, it may be necessary for a tool to implement logic that compares file system metadata and determines whether a non-allocated file has been overwritten by a newer file.

When the beginning of a non-allocated file has been overwritten by a small file, recovery might still be feasible by grafting a generic file header onto the original, Partially Recovered file. This scenario could be sub-classified as Header Repair and could lead to Partially Recovered files being reclassified as Potentially Recovered or even Fully Recovered, depending on the context.

When a file is marked as overwritten by a digital forensic tool, one should recall that the original content of a prior non-allocated file could still be recoverable if the new file was zero bytes in size or incomplete file initialization occurred without any data being overwritten on the acquired storage media [14].

When a Partially Recovered file has been overwritten, it could be sub-classified as Partially Overwritten. Consider the example of an SDCard with

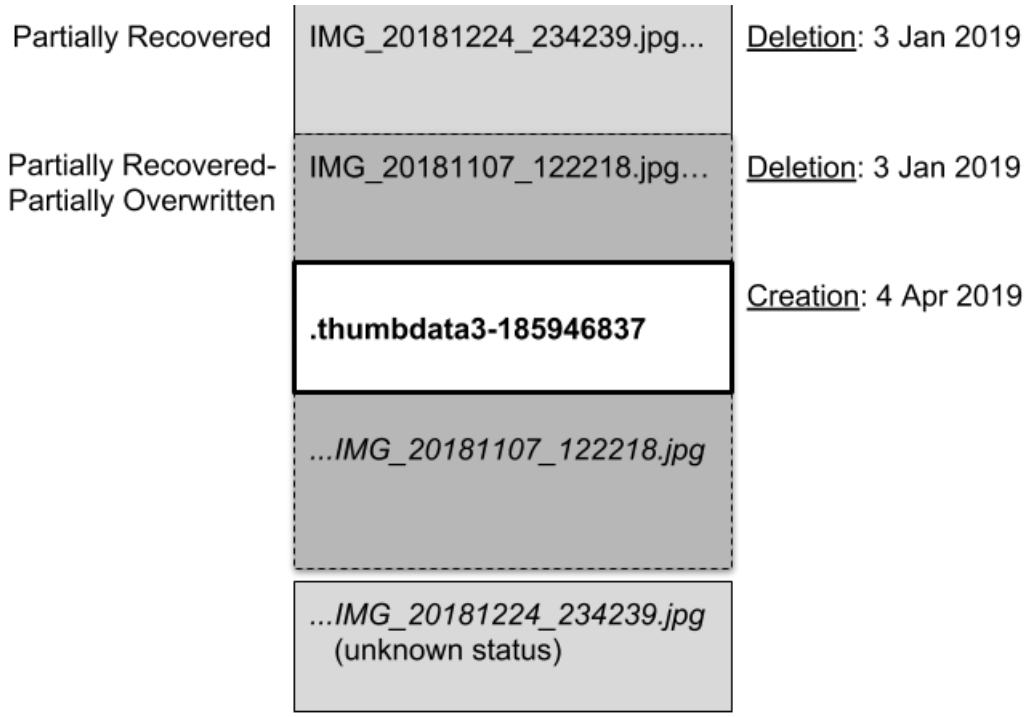

Figure 3: Two non-allocated photographs on an SDCard, one Partially Recovered and the other Partially Recovered with a sub-classification of Partially Overwritten

two deleted photographs in Fig. 3. One photograph was created on 11 November 2018 and saved in contiguous clusters (not fragmented), The other photograph was created on 24 December 2018 (Christmas Eve) with its first fragment saved in clusters immediately preceding the 11 November 2018 photograph, and its remaining content saved somewhere else on the SDCard. After both files are deleted on 3 January 2019, a new file overwrites part of the 11 November 2018 photograph. In this scenario, the 11 November 2018 photograph can be rendered visible despite the overwritten area, and can be classified as Partially Recovered with a sub-classification of Partially Overwritten. Only the beginning of the Christmas Eve photograph can be rendered visible, so it can be classified as Partially Recovered, and the status of the missing portion is unknown.

It is important to note, when all of the content of a non-allocated file has been overwritten, it should be classified as Metadata and Name Recovered.

5.4. Name and Content Recovered

Results of data recovery operations are classified as Name and Content Recovered when content is recovered and can be associated with a recovered name. For instance, a recovered NTFS alternate data stream can have an associated name and content, but does not have its own metadata. As another example, when carving recovered a JPEG data structure with Exif metadata containing a created date of 22 Jan 2019 at 3:43:21PM, and the name `IMG_20190122_154312.JPG` was found in an EXT directory file (no inode recovered), a forensic tool or forensic practitioner associates the recovered content with the name, but without inode metadata. A sample standardized representation of this class of file is provided in Listing 5 of Appendix B, as discussed further in Section 9.

5.5. Metadata and Content Recovered

Results of data recovery operations are classified as Metadata and Content Recovered when content is recovered and can be associated with a recovered file system metadata but no name. For instance, when carving recovered a JPEG data structure with Exif metadata containing a created date of 22 Jan 2019 at 3:43:21PM, and the metadata was found in an EXT inode (no name recovered). A forensic tool or forensic practitioner might associate the recovered content with the metadata, but without the name. Combination of inode analysis and digital stratigraphy could allow association of recovered metadata with content [15]. A sample standardized representation of this

class of file is provided in Listing 6 of Appendix B, as discussed further in Section 9.

5.6. Content Recovered

Results of data recovery operations are classified as Content Recovered when content is recovered, but the associated name and metadata are not recovered. For example, this situation arises when content of digital photographs are recovered using carving methods. When cryptographic hashes (e.g., SHA256) are used to determine that results of carving operations contain known child pornography, it could be asserted to be the complete content of the file with some level of probability. A sample standardized representation of this class of file is provided in Listing 7 of Appendix B, as discussed further in Section 9.

The carving process may recover data structures that where never files in a file system, such as a picture embedded within in a document or database. When the original content of a file can be completely reconstructed from non-contiguous fragments, it should be sub-classified as Fragmented.

When only part(s) of a file's original content have been recovered, the result is not the actual file and is more accurately described as salvaged content [16]. In this case, when only portions of the original file content can be salvaged, it should be sub-classified as Partially Salvaged.

5.7. Metadata and Name Recovered

A file is classified as Metadata and Name Recovered when the file name and metadata can be recovered from a source that never includes information about the original file content (no reference to where the file content was stored on disk). As noted in the introduction of this paper, this situation applies to Windows Shortcut LNK files and Android `external.db` files that contain metadata for files stored on external or encrypted media. This situation can also arise when recovering information about files and folders from an INDX entry or the `$LogFile` on NTFS [17]. For example, Fig. 4 shows the directory `dir3` that had its MFT entry overwritten by a new file, but a remnant entry was recovered from the `$LogFile`. Fig. 4 describes the `dir3` entry as "file contents unknown" because the information recovered from `$LogFile` does not include a reference to the contents on disk. A sample standardized representation of this class of file is provided in Listing 8 of Appendix B, as discussed further in Section 9.

starting at a child and up through parent directories, to the current file system root. While Fig. 5 illustrates at the granularity of file updates, these updates can be done at the granularity of block and indirect block pointers, or single directory BTree nodes. An early illustration of this update system can be seen in a publication on WAFL [19, Figures 1–4].

Versioned file systems simplify recoverability in some cases, particularly if the data being recovered are included in the snapshot history. If a file is seen to exist intact in a prior snapshot but not the current file system state, it can be classified as Fully Recovered. Versioned file systems impact the other recovery states with supplemental partial-state data. The frequent recording of consistent file system structures leaves residual artifacts that in some file systems can be stratigraphically [14] related to one another, such as with BTRFS's transids that guarantee a generational ordering [22]. This frequent recording also leaves little incentive for a versioned file system to record a reference to a non-allocated file. Instead, the next version of the file or containing directory can be written without that file reference, 5–30 seconds from the unallocation operation. Overall, tools' assessments of recoverability will be tied to whether content can be related to data structures found in—or if on-disk layout can be utilized, between—active superblocks and their descendent allocation layouts.

7.2. Recoverability in SQLite databases

To assess the generality of the proposed classification of file recovery, here it is applied to SQLite databases, which resemble a file system within a file. Specifically, in the B-Tree structure of SQLite databases, the root page is analogous to the top level directory of a file system, the internal pages and leaf pages are analogous to the subdirectories, and cells are analogous to files, with the body of each record analogous to content. Overflow pages are analogous to fragmented files with additional content. The SQLite WAL file is analogous to those in versioning file systems.

When a row is unallocated within a SQLite database, the space that the row occupies is marked as free, and part of the row header is overwritten. Such recovered rows in free space can be treated as Partially Recovered but could subsequently be authenticated as Fully Recovered. In

addition, multiple rows in a SQLite database can become non-allocated when a complete leaf page is unallocated. In this situation, the rows remain Potentially Recovered (or Fully Recovered if determined by authentication) in non-allocated areas within a SQLite file until they are overwritten by new rows. For such recovery results, the **name** of the associated table can be determined.[6] Rows recovered from within a SQLite database without associated table name can be classified as Metadata and Content Recovered.

Some tools do not distinguish between a non-allocated row that is Potentially Recovered but requires authentication, versus a Fully Recovered row that has been authenticated either automatically or by the user.

Rows that can be carved from unallocated space within a SQLite database in their entirety, and the table name can be determined, can be classified as Name and Content Recovered. Rows with only content partially carved from unallocated space, without the ability to determine the table name, can be classified as Content Recovered.

When a page associated with a table contains empty rows or a gap in ROWID, a tool or forensic practitioner might assert that rows were non-allocated and auto vacuum removed the content. It is important to note that if the developer used Vacuum instead of Auto Vacuum, the blank row is not maintained but there will be a gap in the ROWID sequence. These recovery results could be classified as Metadata and Name Recovered. For example, the following query result from a WhatsApp database (msgstore.db with auto vacuum fully enabled) shows a gap between the second to last row (ROWID 15) and last row (ROWID 21).

```
$ sqlite3 msgstore.db "SELECT _id,key_remote_jid,key_from_me,
data,timestamp FROM messages"
1|-1|0||0
2|447937961169@s.whatsapp.net|1||1481645867714
3|447937961169@s.whatsapp.net|0|High festivity...|1481645868000
4|447937961169@s.whatsapp.net|0||1481645869000
```

[6]Notably, when analysing SQLite databases, it might be necessary to correlate information from multiple table names.

```
5|447937961169@s.whatsapp.net|1|Indeed but we ...1481647150508
6|447937961169@s.whatsapp.net|1|Festive but past security|1481647171472
7|447937961169@s.whatsapp.net|1||1481647184368
8|447937961169@s.whatsapp.net|0|Parfait!|1481647815000
9|447937961169@s.whatsapp.net|1|They stopped me ...|1481649923395
10|447937961169@s.whatsapp.net|1||1481649954060
11|447937961169@s.whatsapp.net|0|I made it - stay calm...|1481650025000
12|447937961169@s.whatsapp.net|1||1481650118983
13|447937961169@s.whatsapp.net|1|Are you sure?|1481650128186
14|447937961169@s.whatsapp.net|0|I see you...|1481650325000
15|447937961169@s.whatsapp.net|1|Whew!|1481650442187
21|447937961169@s.whatsapp.net|1|Wow!|1481707272518
```

SQLite databases can contain index tables with partial information about referenced rows stored in another table. The row in the index table could be considered as Partially Recovered if the table name can be determined, or Content Recovered if the table name is unknown.

Complexities arise in SQLite version 3.7.0 and later which use a Write Ahead Log ("WAL"'). The WAL file can contain a new row that has not yet been committed to the database. Therefore, it would be incorrect to describe this row "deleted," but some tools do just that. Furthermore, the WAL file can simultaneously contain the first instance of a row, as well as updated copies of the row, and a final copy when a row becomes non-allocated [23].

Therefore, rows in a WAL file should only be described as "deleted" (meaning non-allocated) when there is a clear progression of earlier instances, and the final state is that the row is no longer in the database.

> *"From the above descriptions, it should be clear that there are many instances in which a record can be recovered which may be a copy of a live record. Therefore, in these instances, it is important not to call any recovered record a "deleted record" and compare recovered records with the set of live records, currently within a database, to determine whether the record is, in fact, a copy (or partial copy) of a live record or indeed is deleted."* [24]

Under these circumstances, although the original row in the database has been unallocated and overwritten, its complete content can be recovered from the WAL file. Therefore, the row in the WAL file can be classified as Fully Recovered.

Table 3: Examples of recoverability classification applied to SQLite databases

Class	**SQLite**
Fully Recovered	Row is recovered from the database and/or WAL file and all parts correspond to the original schema
Potentially Recovered	Row is recovered but has not been authenticated, or some parts are missing or cannot be matched with a schema
Partially Recovered	Metadata for a row are recovered, but the original content of the row is allocated to a new row
Name and Content Recovered	Some information is recovered using data salvaging methods and the table name can be determined
Metadata and Content Recovered	Metadata and content for a row are recovered, but the associated table name is not known
Content Recovered	Some content is recovered using data salvaging methods but no associated metadata and table name can be determined
Metadata and Name Recovered	Some metadata is recovered and the table name can be determined
Metadata Recovered	Metadata for a row are recovered, but the original content of the row is not known

SQLite recovery scenarios might not fit the Name Recovered class, where only table name is recovered without any other information.

8. Prior Representation of File Allocation Status

The precursor question to whether a file is recoverable is, in a simplified form, first whether the file is allocated. Digital Forensics XML (DFXML), a language that describes storage system forensics, has previously described a file's allocation status as a combination of the allo-

cation of the name structure and metadata structure.[7] The taxonomy proposed in this paper draws on this granular allocation reporting, but the nuance of labeling recovered content highlights improvements that could be made to DFXML. For instance, at the time of this writing, the DFXML vocabulary does not have a term for denoting a file's byte runs as being complete according to a data structure parse. That is, a tool could be interrupted before finishing parsing (e.g., a B-Tree), but there is no vocabulary in DFXML to denote that the parse was only partially successful. A proposal[8] to further report the geometry of file systems via reporting the on-disk addresses of inode and directory entries has not yet been realized with a sample implementation, but would be a boon to recognizing when regions of a disk have been shared between allocated and unallocated files, assisting with designating the recoverability of a file as Partially Recovered.

For instance, suppose a large file, such as a virtual machine disk image, is found in a non-allocated state, with all data structures intact. Absent context of the rest of the file system, this file could be classified as Fully Recovered. However, if further forensic examination finds that portions of this large file have been overwritten by another file, then the the virtual machine disk image could be reclassified as Partially Recovered. Further, this reclassification would be detected more faithfully (and possibly draw on digital stratigraphy analysis) by a tool incorporating the addresses of inodes. In the particular case of a virtual disk image file, analysis of the inodes could be key to a forensic practitioner distinguishing guest virtual machine file system structures from the host machine's file system structures.

Appendix A presents a mapping of DFXML that implements the UnallocatedRecoverability classifications for any tool that generates DFXML with granular allocation reporting, with some allowance made for features brought to light or reinforced by this paper.

[7] https://github.com/dfxml-working-group/dfxml_schema/issues/14
[8] https://github.com/dfxml-working-group/dfxml_schema/issues/5

9. Standard Representation of File Recovery Results

The community-driven initiative called CASE (caseontology.org), that supports standardized representation, interoperability and automation in cyber-investigations,has a property bundle named `UnallocatedRecoverability` for representing the recoverability of non-allocated files [4]. Table 4 lists the property names and associated values in the `UnallocatedRecoverability` property bundle. The next section will relate this property bundle to the taxonomy of Table 1.

Table 4: Property names and associated values in the CASE UnallocatedRecoverability property bundle

Values	Property
recovered overwritten unknown	nameStatus
recovered overwritten unknown	metadataStatus
recovered overwritten unknown	contentStatus

The results of file recovery operations can be represented explicitly using CASE. Recovered metadata are represented using the File property bundle in CASE, including `fileName` and `filePath`, and additional information about a file are represented using the `MftRecord` and `ExtInode` property bundles. In addition, the location of a recovered file in a specific data source is represented using the `PathRelation` property bundle on a "contained-within" `Relationship` object (e.g., targeting a specified `DiskPartition` trace). Similarly, recovered content is represented in CASE using the `DataRange` property bundle on a "contained-within" `Relationship` object.

Examples of the standard CASE representation for each class of file recovery are provided in Appendix B.

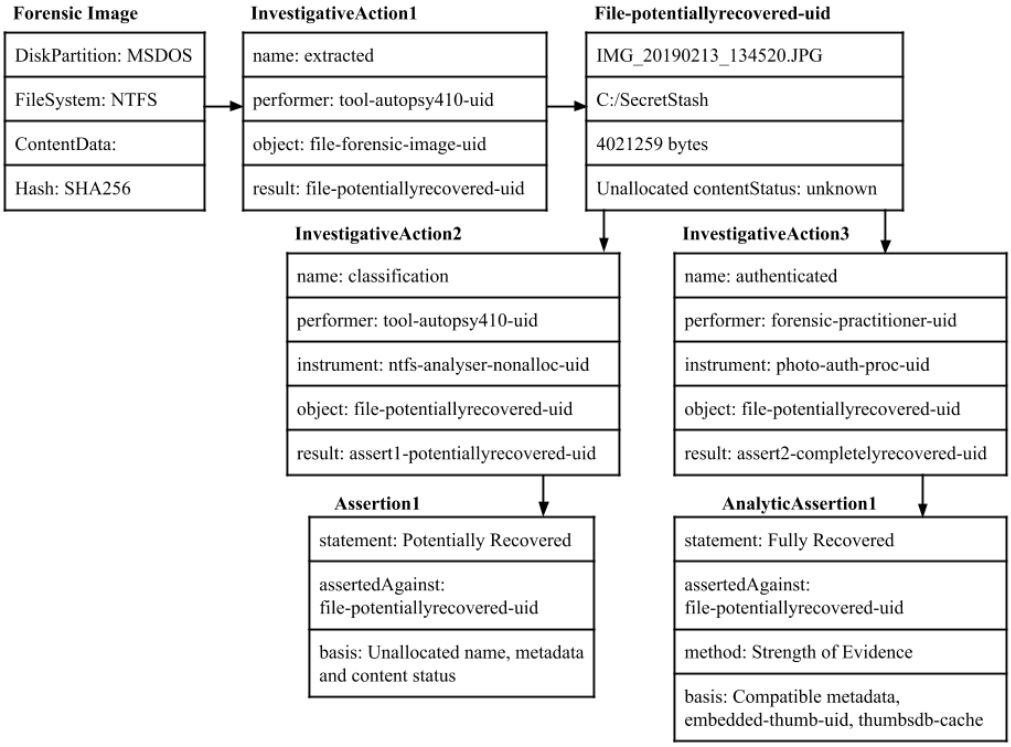

Figure 6: Authentication of file that was recovered with unknown content status, resulting in an `AnalyticAssertion` that the file is Fully Recovered.

10. Standard Representation of File Recovery Results

When the results of file recovery operations are classified, either automatically using digital forensic software or manually by a forensic practitioner, the authentication and classification decisions can be represented using CASE. For illustrative purposes, imagine that a forensic tool extracted a non-unallocated file and then performed an automated process on the basis of NTFS file system metadata, resulting in an assertion that the file was `Potentially Recovered`. Then a forensic practitioner applied an authentication process to the `Potentially Recovered` file in Listing 3 in Appendix B and decided that it was `Fully Recovered`. Fig. 6 illustrates the authentication process applied to a file that was recovered with unknown content status, resulting in an assertion that the file is `Fully Recovered`.

In CASE, this authentication process can be represented as an `InvestigativeAction` and the resulting decision i.e., the (re)classification of the file as Fully Recovered, can be represented with using an `AnalyticAssertion` object as shown in Listing 1 below.

The `AnalyticAssertion` object in Listing 1 specifies that a strength of evidence approach was used on the basis of forensic analysis of the metadata, content and compatible thumbnail data. In this example, the forensic practitioner has the opinion, with 90% confidence, that the evidence has a 95% probability under the assertion that the file is Fully Recovered.

Listing 1: Authentication process applied to the Potentially Recovered file in Listing 3 and changing its classification to Fully Recovered, represented using CASE in JSON-LD.

```
{
  "@id": "investigative-action1-45a5ec72-18b2-4f1c-
     c325-2c4b26a64fe3",
  "@type": "InvestigativeAction",
  "name": "authenticated",
  "startTime": "2019-03-17T12:01:23.14Z",
  "propertyBundle": [
    {
      "@type": "ActionReferences",
      "instrument": "authentication-process1-uuid",
      "location": "forensic-laboratory1-uuid",
      "performer": "forensic-practitioner1-uuid",
      "object": [
        "provenance-record1-uuid",
          "file-potentiallyrecovered-38e5cd74-19b2-3
             f0c-b324-1c4b25a34f12"
      ],
      "result": [
        "provenance-record2-uuid",
          "analytic-assertion-fullyrecovered-38
             e5cd74-19b2-3f0c-b324-1c4b25a34f12"
      ]
    }
  ]
},
{
```

```
  "@id": "analytic-assertion-fullyrecovered-uuid",
  "@type": "AnalyticAssertion",
  "statement": "Fully Recovered",
  "assertedAgainst": [file-potentiallyrecovered-38
     e5cd74-19b2-3f0c-b324-1c4b25a34f12],
  "method": "strength of evidence",
  "probability": 95,
  "confidence": 90,
  "rationale": [embedded-thumbnail-uid, thumbdb-
     cache12-uid]
}
```

11. Conclusion and Future Work

Applying classification and authentication to file recovery, and standardizing how these files are represented, increases the clarity and consistency of how results of file recovery operations are treated in digital forensic science. Such consistency helps digital forensic practitioners understand the context and reliability of file recovery operations, helping them reason about file recovery results and reducing the chances of mistakes.

A software developer can create a competitive advantage by satisfying such standardized requirements. Ultimately, demonstrating that a tool meets the defined requirements could become a prerequisite for use of tools in accredited forensic service providers, under the ISO/IEC 27041 standard. In addition to putting themselves in a strong position to meet standards related to digital forensics, software developers who provide verification documentation along with their tool will greatly enhance subsequent tool testing efforts. Comprehensively tested software helps find and fix bugs, reduces the risk of errors reaching the courtroom, and increases the trust in digital forensics as a discipline [3]. The NIST Computer Forensic Tool Testing (CFTT) program developed the Federated Testing Project, to allow for more widespread testing of digital forensic tools by outside forensics laboratories. Nelson et al. demonstrated a method to compare results of multiple tools by using a common structured output [25]. Standardizing the classification of file recovery results and representing results using CASE also enables

interoperability between tools/systems, and automated normalization, combination, correlation, and validation of information for analysis and tool testing purposes.

In the future, tools that perform file recovery operations could include a function that enables a forensic practitioner to change the classification of file recovery results on the basis of their own authentication result. As part of this functionality, tools could allow forensic practitioners to represent the reclassification in a structured form, such as an assertion with associated confidence, that is added to the case file.

The inaugural set of requirements for classifying file recovery results is provided as a foundation for building consensus in the digital forensic community. Any such requirements that uses the taxonomy defined in this work will need to be updated as new file systems, recovery methods and knowledge emerge. These requirements could be established as a standard and could be maintained by an independent organization, providing stability and versioning, to serve the satisfy the needs of all stakeholders in the digital forensic community, including tool developers, fact-finders and decision-makers.

The Assertion object proposed in this paper is undergoing review by the CASE and UCO community.

12. Acknowledgements

The authors thank Sean Barnum for his assistance with representing assertions in CASE supported by the Unified Cyber Ontology (UCO). We also thank Hannes Spichiger at University of Lausanne, and the peer reviewers for Digital Investigation who provided many insightful recommendations that improved this paper.

Appendix A. Implementation of Classification via Translation from DFXML

This Appendix describes an implementation of the UnallocatedRecoverability classification and sub-classification taxonomy, via translation from DFXML. Table A.5 reports the mapping, with these columns describing DFXML-sourced information:

1. Inode - the status of the inode-like structure (e.g., Linux EXT inode, MFT entry) of the file, either "alloc" for allocated, "unalloc" for present but unallocated, or "absent" for unknown.
2. Name - the status of the name-like structure (e.g., directory entry) of the file, with the same values as for the Inode column.
3. Byte runs - the status of the byte runs list that describe the file's content, either "full" for fully found, "partial" for partially found, or "absent." Note that "partial" stemmed from the development of the taxonomy in this paper, so it has not yet been implemented in as a vocabulary element of DFXML.
4. Overlap - Whether the content byte runs (when present) overlap with any other file or data structure in the file system, such as would happen in an overwrite. To account for files with multiple hard links, "other file" refers to files with different inodes.

The descriptions in the table are assumed to apply to files of greater than 0 bytes in size, because some matters are trivialized for 0-byte files. Also, some combinations of the first four columns describe file system states that are inconsistent or damaged, but they are mapped here to account for all possible states that forensic practitioners will encounter. Combinations that are theoretically impossible are designated "N/A" in the mapping.

Table A.5: Mapping of DFXML to UnallocatedRecoverability classification and sub-classification properties.

Inode	**Name**	**Byte runs**	**Overlap**	**Classification**	**Sub-classification**
alloc	alloc	full	no	fully recovered	
alloc	alloc	full	yes	partially recovered	partially overwritten
alloc	alloc	partial	no	potentially recovered	fragmented
alloc	alloc	partial	yes	partially recovered	partially overwritten
alloc	alloc	absent	N/A	metadata and name recovered	
alloc	unalloc	full	no	fully recovered	
alloc	unalloc	full	yes	partially recovered	partially overwritten
alloc	unalloc	partial	no	potentially recovered	fragmented
alloc	unalloc	partial	yes	partially recovered	partially overwritten
alloc	unalloc	absent	N/A	metadata and name recovered	
alloc	absent	full	no	metadata and content recovered	path unrecovered
alloc	absent	full	yes	partially recovered	partially overwritten AND path unrecovered
alloc	absent	partial	no	potentially recovered	path unrecovered
alloc	absent	partial	yes	partially recovered	partially overwritten AND path unrecovered
alloc	absent	absent	N/A	metadata recovered	path unrecovered
unalloc	alloc	full	no	fully recovered	
unalloc	alloc	full	yes	partially recovered	partially overwritten
unalloc	alloc	partial	no	potentially recovered	
unalloc	alloc	partial	yes	partially recovered	partially overwritten
unalloc	alloc	absent	N/A	metadata and name recovered	
unalloc	unalloc	full	no	fully recovered	
unalloc	unalloc	full	yes	partially recovered	partially overwritten
unalloc	unalloc	partial	no	potentially recovered	fragmented
unalloc	unalloc	partial	yes	partially recovered	partially overwritten
unalloc	unalloc	absent	N/A	metadata and name recovered	
unalloc	absent	full	no	metadata and content recovered	path unrecovered
unalloc	absent	full	yes	partially recovered	partially overwritten AND path unrecovered
unalloc	absent	partial	no	potentially recovered	path unrecovered
unalloc	absent	partial	yes	partially overwritten	partially overwritten AND path unrecovered
unalloc	absent	absent	N/A	metadata recovered	
absent	alloc	full	no	name and content recovered	
absent	unalloc	partial	no	name and content recovered	
absent	alloc	full	yes	name and content recovered	fragmented
absent	unalloc	partial	yes	name and content recovered	fragmented
absent	alloc	N/A	N/A	name recovered	
absent	unalloc	N/A	N/A	name recovered	
absent	absent	full	no	content recovered	path unrecovered
absent	absent	full	yes	content recovered	fragmented AND path unrecovered
absent	absent	partial	no	content recovered	fragmented AND path unrecovered
absent	absent	partial	yes	content recovered	fragmented AND path unrecovered
absent	absent	absent	N/A	N/A	

The end of the table denotes the possibility that a file's data addresses are asserted to be recovered without any associated inode or directory entry. These examples are provided as a reminder that recovery of a file can make use of supplementary data sources outside of the subject media's file system structures. One example would be a prior snapshot of a non-snapshotting file system being known to contain the data of interest. Another would be sector-level hashes that either individually, or in co-location, are asserted to uniquely indicate a file ([26], Section 3)."

Appendix B. Examples of CASE representation of File Recovery Classifications

This Appendix provides examples of the standard CASE representation for each class of file recovery:

– Listing 2: Fully Recovered JPG file
– Listing 3: Potentially Recovered JPG file
– Listing 4: Partially Recovered JPG file
– Listing 5: Name and Content Recovered SQLite file
– Listing 6: Metadata and Content Recovered SQLite file
– Listing 7: Content Recovered PDF file
– Listing 8: Name and Metadata Recovered DOCX file
– Listing 9: Metadata Recovered
– Listing 10: Name Recovered MP4 file

These examples are abstracted from real world instances to demonstrate how file recovery results can be represented in a normalized form. Representation of specific methods used by individual tools to make assertions on classification and authentication is an area of future work.

Listing 2: Example of a Fully Recovered JPG file represented using CASE in JSON-LD, showing an NTFS file system with an intact MFT entry linked to the original file content.

```
{
      "@id": "file-fullyrecovered-38e5cd74-19b2-3f0c
         -b324-1c4b25a34f12",
      "@type": "Trace",
```

```
"propertyBundle": [
  {
    "@type": "File",
    "createdTime": "2019-01-22T15:43:13.32Z",
    "extension": "jpg",
    "fileName": "IMG_20190122_154312.JPG",
    "fileSystemType": "NTFS",
    "filePath": "C:/SecretStash/
       IMG_20190122_154312.JPG",
    "isDirectory": false,
    "allocationStatus": "unallocated",
    "sizeInBytes": 4021529
  },
  {
    "@type": "UnallocatedRecovery",
    "nameStatus": "recovered",
    "metadataStatus": "recovered",
    "contentStatus": "recovered",
  },
  {
    "@type": "MftRecord",
    "mftFileID": "532552",
    "mftRecordChangeTime": "2019-01-22T15
       :43:13.32Z",
    "mftFileNameCreatedTime": "2019-01-22T15
       :43:13.32Z",
  },
  {
    "@type": "ContentData",
    "magicNumber": "/9j/4Sn+RXhpZg",
    "mimeType": "image/jpg"
    "hash": [
      {
        "@type": "Hash",
        "hashMethod": "SHA256",
        "hashValue": "1
           f635be55c83a3dff3c771f4b4b36202f
           79d4bc0c109bd83d9609cf45a47b22d"
      }
    ],
```

```
        }
      ]
},
{
      "@id": "datarange-relationship1",
      "@type": "Relationship",
      "source": "file-fullyrecovered-38e5cd74-19b2-3
         f0c-b324-1c4b25a34f12",
      "target": "diskpartition1-46d3ae54-23a4-2e1a-
         a563-2c4b25a35d36",
      "kindOfRelationship": "contained-within",
      "isDirectional": true,
      "propertyBundle": [
        {
          "@type": "DataRange",
          "rangeOffset": 5635584,
          "rangeSize": 4021529
        }
      ]
},
{
      "@id": "filepath-relationship1",
      "@type": "Relationship",
      "source": "file-fullyrecovered-38e5cd74-19b2-3
         f0c-b324-1c4b25a34f12",
      "target": "diskpartition1-46d3ae54-23a4-2e1a-
         a563-2c4b25a35d36",
      "kindOfRelationship": "contained-within",
      "isDirectional": true,
      "propertyBundle": [
        {
          "@type": "PathRelation",
          "path": "C:/SecretStash/
             IMG_20190122_154312.JPG"
        }
      ]
}
```

Listing 3: Example of a Potentially Recovered JPG file on an NTFS file system represented using CASE in JSON-LD.

```
{
  "@id": "file-potentiallyrecovered-38e5cd74-19b2-3
     f0c-b324-1c4b25a34f12",
  "@type": "Trace",
  "propertyBundle": [
    {
      "@type": "File",
      "createdTime": "2019-02-13T13:45:21.49Z",
      "extension": "jpg",
      "fileName": "IMG_20190213_134520",
      "fileSystemType": "NTFS",
      "filePath": "C:/SecretStash/
         IMG_20190213_134520.JPG",
      "isDirectory": false,
      "allocationStatus": "unallocated",
      "sizeInBytes": 4021529
    },
    {
      "@type": "UnallocatedRecovery",
      "nameStatus": "recovered",
      "metadataStatus": "recovered",
      "contentStatus": "unknown",
    },
    {
      "@type": "MftRecord",
      "mftFileID": "532552",
      "mftRecordChangeTime": "2019-02-13T13:45:21.49
         Z",
      "mftFileNameCreatedTime": "2019-02-13T13
         :45:21.49Z"
    },
    {
      "@type": "ContentData",
      "magicNumber": "/9j/4Sn+RXhpZg",
      "mimeType": "image/jpg"
      "hash": [
        {
          "@type": "Hash",
          "hashMethod": "SHA256",
          "hashValue": "5
```

```
            f635be55c83a3dff3c771f4b4b36202f79
            d4bc0c109bd83d9609cf45a47b23c"
        }
      ],
    }
  ]
},
{
  "@id": "datarange-relationship2",
  "@type": "Relationship",
  "source": "file-potentiallyrecovered-38e5cd74-19b2
     -3f0c-b324-1c4b25a34f12",
  "target": "diskpartition1-46d3ae54-23a4-2e1a-a563
     -2c4b25a35d36",
  "kindOfRelationship": "contained-within",
  "isDirectional": true,
  "propertyBundle": [
    {
      "@type": "DataRange",
      "rangeOffset": 5635584,
      "rangeSize": 4021529
    }
  ]
},
{
  "@id": "filepath-relationship2",
  "@type": "Relationship",
  "source": "file-potentiallyrecovered-38e5cd74-19b2
     -3f0c-b324-1c4b25a34f12",
  "target": "diskpartition1-46d3ae54-23a4-2e1a-a563
     -2c4b25a35d36",
  "kindOfRelationship": "contained-within",
  "isDirectional": true,
  "propertyBundle": [
    {
      "@type": "PathRelation",
      "path": "C:/SecretStash/IMG_20190213_134520.
         JPG"
    }
  ]
```

```
}
```

Listing 4: Example of a Partially Recovered JPG file represented using CASE in JSON-LD, showing an NTFS files system with an intact MFT entry but overwritten content.

```
{
  "@id": "file-overwritten-38e5cd74-19b2-3f0c-b324-1
     c4b25a34f12",
  "@type": "Trace",
  "propertyBundle": [
      {
        "@type": "File",
        "createdTime": "2019-02-30T19:27:43.28Z",
        "extension": "jpg",
        "fileName": "IMG_20190230_274326.JPG",
        "fileSystemType": "NTFS",
        "filePath": "C:/SecretStash/
           IMG_20190230_274326.JPG",
        "isDirectory": false,
        "allocationStatus": "unallocated",
        "sizeInBytes": 4142567
  },
  {
        "@type": "UnallocatedRecovery",
        "nameStatus": "recovered",
        "metadataStatus": "recovered",
        "contentStatus": "overwritten",
  },
  {
        "@type": "MftRecord",
        "mftFileID": "646210",
        "mftRecordChangeTime": "2019-02-30T19
           :27:43.28Z",
        "mftFileNameCreatedTime": "2019-02-30T19
           :27:43.28Z",
  },
  {
        "@type": "ContentData",
        "hash": [
          {
```

```
            "@type": "Hash",
            "numberHashes": "0",
          }
    ],
  }
 ]
},
{
  "@id": "datarange-relationship3",
  "@type": "Relationship",
  "source": "file-partiallyrecovered-38e5cd74-19b2-3
     f0c-b324-1c4b25a34f12",
  "target": "diskpartition1-46d3ae54-23a4-2e1a-a563
     -2c4b25a35d36",
  "kindOfRelationship": "contained-within",
  "isDirectional": true,
  "propertyBundle": [
      {
        "@type": "DataRange",
        "rangeOffset": 8931840,
        "rangeSize": 4142567
  }
 ]
},
{
  "@id": "filepath-relationship3",
  "@type": "Relationship",
  "source": "file-overwritten-38e5cd74-19b2-3f0c-
     b324-1c4b25a34f12",
  "target": "diskpartition1-46d3ae54-23a4-2e1a-a563
     -2c4b25a35d36",
  "kindOfRelationship": "contained-within",
  "isDirectional": true,
  "propertyBundle": [
      {
        "@type": "PathRelation",
        "path": "C:/SecretStash/IMG_20190230_274326.
           JPG"
  }
 ]
```

```
},
```

Listing 5: Example of a SQLite file with Name and Content Recovered represented using CASE in JSON-LD, with name and content recovered but without metadata.

```
{
   "@id": "file-namecontentrecovered-24a6cd42-19b2-3
      f0c-b324-1c4b25b56a24",
   "@type": "Trace",
   "propertyBundle": [
    {
          "@type": "File",
          "extension": "sql",
          "fileName": "msgstore.db",
          "fileSystemType": "EXT",
          "allocationStatus": "unallocated"
    },
    {
          "@type": "ContentData",
          "magicNumber": "U1FMaXRlIGZvcm1hdCAzCg==",
          "mimeType": "application/x-sqlite3"
          "hash": [
            {
              "@type": "Hash",
              "hashMethod": "SHA256",
              "hashValue": "
                 ad40b76749ec5ebc015b25c15c2e0d628e7
              c5fd7d8b03cab854ddc5e27304b51"
            }
          ],
          "dataPayload": "<content cut for brevity
             >",
          "sizeInBytes": 10768384
    },
    {
          "@type": "UnallocatedRecovery",
          "nameStatus": "recovered",
          "metadataStatus": "overwritten",
          "contentStatus": "recovered",
    },
  ]
```

```
},
{
   "@id": "filepath-relationship6",
   "@type": "Relationship",
   "source": "forensicduplicate-3442ca12-25b3-5f1c-
      d334-2c432595bc13",
   "target": "file-contentrecovered-24a6cd42-19b2-3
      f0c-b324-1c4b25b56a24",
   "kindOfRelationship": "contained-within",
   "isDirectional": true,
   "propertyBundle": [
    {
          "@type": "DataRange",
          "rangeOffset": 2322406,
          "rangeSize": 10768384
    }
  ]
},
```

Listing 6: Example of a SQLite file with Metadata and Content Recovered represented using CASE in JSON-LD, with metadata and content recovered and associated but without the associated name.

```
{
   "@id": "file-metadatacontentrecovered-24a6cd42-19
      b2-3f0c-b324-1c4b25b56a24",
   "@type": "Trace",
   "propertyBundle": [
    {
      "@type": "ExtInode",
      "extInodeID": "124596",
      "extInodeChangeTime": "2018-11-07T12:08:34.02Z
         ",
      "extPermissions": "4755",
      "extInodeSUID": "1005"
    },
    {
          "@type": "ContentData",
          "magicNumber": "U1FMaXRlIGZvcm1hdCAzCg==",
          "mimeType": "application/x-sqlite3"
          "hash": [
```

```
            {
              "@type": "Hash",
              "hashMethod": "SHA256",
              "hashValue": "5
                 c8708886819985b437176b27cd4d49c0d6
              b5b641c5d0546112d8363e0bcd28a"
            }
          ],
          "dataPayload": "<content cut for brevity
             >",
          "sizeInBytes": 10768384
    },
    {
          "@type": "UnallocatedRecovery",
          "nameStatus": "null",
          "metadataStatus": "recovered",
          "contentStatus": "recovered",
    },
  ]
},
{
   "@id": "filepath-relationship6",
   "@type": "Relationship",
   "source": "forensicduplicate-3442ca12-25b3-5f1c-
      d334-2c432595bc13",
   "target": "file-metadatacontentrecovered-24a6cd42
      -19b2-3f0c-b324-1c4b25b56a24",
   "kindOfRelationship": "contained-within",
   "isDirectional": true,
   "propertyBundle": [
    {
          "@type": "DataRange",
          "rangeOffset": 2323578,
          "rangeSize": 10768384
    }
  ]
},
```

Listing 7: Example of a PDF file with Content Recovered represented using CASE in JSON-LD, with content recovered but without a name or link to the associated file

content (e.g., using carving methods on a forensic copy/duplicate of a data source).


```
{
   "@id": "file-contentrecovered-24a6cd42-19b2-3f0c-
      b324-1c4b25b56a24",
   "@type": "Trace",
   "propertyBundle": [
   {
          "@type": "ContentData",
          "magicNumber": "JVBERi0xLjMKJQ",
          "mimeType": "application/pdf"
          "hash": [
            {
              "@type": "Hash",
              "hashMethod": "SHA256",
              "hashValue": "6
                 affd27d8e3bcab5805f168aa5c8015b7bb
              f7779a69530b3ba2fe78f2d48b2a6"
            }
          ],
          "dataPayload": "<content cut for brevity
             >",
          "sizeInBytes": 4734559
    },
    {
          "@type": "UnallocatedRecovery",
          "nameStatus": "overwritten",
          "metadataStatus": "overwritten",
          "contentStatus": "recovered",
    },
  ]
},
{
   "@id": "filepath-relationship5",
   "@type": "Relationship",
   "source": "forensicduplicate-3442ca12-25b3-5f1c-
      d334-2c432595bc13",
   "target": "file-contentrecovered-24a6cd42-19b2-3
      f0c-b324-1c4b25b56a24",
   "kindOfRelationship": "contained-within",
   "isDirectional": true,
```

```
   "propertyBundle": [
    {
          "@type": "DataRange",
          "rangeOffset": 2322406,
          "rangeSize": 4734559
    }
  ]
},
```

Listing 8: Example of a DOCX file with Metadata and Name Recovered represented using CASE in JSON-LD, the with name and metadata recovered from an NTFS file system without a link to the associated non-allocated content.

```
{
   "@id": "file-metadatanamerecovered-38e5cd74-19b2
      -3f0c-b324-1c4b25a34f12",
   "@type": "Trace",
   "propertyBundle": [
    {
          "@type": "File",
          "createdTime": "2018-05-22T10:38:34.02Z",
          "extension": "docx",
          "fileName": "Confidential-2018-05-22.DOCX
             ",
          "fileSystemType": "NTFS",
          "filePath": "C:/Sensitive/Confidential
             -2018-05-22.DOCX",
          "isDirectory": false,
          "allocationStatus": "unallocated",
          "sizeInBytes": 4031432
        },
        {
          "@type": "UnallocatedRecoverability",
          "nameStatus": "recovered",
          "metadataStatus": "recovered",
          "contentStatus": "null",
        },
        {
          "@type": "MftRecord",
          "mftFileID": "732615",
          "mftRecordChangeTime": "2018-05-25T10
```

```
            :38:34.02Z",
          "mftFileNameCreatedTime": "2018-05-22T10
            :38:34.02Z"
    }
  ]
},
{
   "@id": "filepath-relationship4",
   "@type": "Relationship",
   "source": "file-metadatanamerecovered-38e5cd74-19
      b2-3f0c-b324-1c4b25a34f12",
   "target": "diskpartition1-46d3ae54-23a4-2e1a-a563
      -2c4b25a35d36",
   "kindOfRelationship": "contained-within",
   "isDirectional": true,
   "propertyBundle": [
     {
       "@type": "PathRelation",
       "path": "C:/Sensitive/Confidential
          -2018-05-22.DOCX"
     }
   ]
},
```

Listing 9: Example of Metadata Recovered represented using CASE in JSON-LD, with metadata recovered from an EXT inode, but without a name or link to the associated file content.

```
{
  "@id": "file-metadatarecovered-24a6cd42-19b2-3f0c-
     b324-1c4b25b56a24",
  "@type": "Trace",
  "propertyBundle": [
    {
      "@type": "File",
      "createdTime": "2018-01-27T10:38:34.02Z",
      "extension": "null",
      "fileName": "null",
      "fileSystemType": "EXT3",
      "filePath": "null",
      "isDirectory": false,
```

```
      "allocationStatus": "unallocated",
      "sizeInBytes": 5123551
    },
    {
      "@type": "UnallocatedRecovery",
      "nameStatus": "overwritten",
      "metadataStatus": "recovered",
      "contentStatus": "unknown",
    },
    {
      "@type": "ExtInode",
      "extInodeID": "124593",
      "extInodeChangeTime": "2018-11-06T10:08:34.02Z
         ",
      "extPermissions": "4755",
      "extInodeSUID": "1005"
    }
  ]
}
```

Listing 10: Example of a Name Recovered MP4 file represented using CASE in JSON-LD, with name recovered but without metadata or a link to the associated file content (e.g., a lone EXT directory entry).

```
{
   "@id": "file-namerecovered-24a6cd42-19b2-3f0c-
      b324-1c4b25b56a24",
   "@type": "Trace",
   "propertyBundle": [
    {
          "@type": "File",
          "extension": "mp4",
          "fileName": "MVI_0022.MP4",
          "fileSystemType": "NTFS",
          "allocationStatus": "unallocated"
    },
    {
          "@type": "UnallocatedRecovery",
          "nameStatus": "recovered",
          "metadataStatus": "overwritten",
          "contentStatus": "overwritten",
```

```
    },
  ]
},
```


[1] NIST, Active file identification & deleted file recovery tool specification, 2009. URL: https://www.cftt.nist.gov/DFR-req-1.1-pd-01.pdf, accessed: 2018-11-13.

[2] UK Forensic Regulator, Method validation in digital forensics, 2015. URL: https://www.gov.uk/government/publications/method-validation-in-digital-f accessed: 2018-11-13.

[3] E. Casey, Cutting the gordian knot: Defining requirements for trustworthy tools, Journal of Digital Investigation 8 (2012) 145–146.

[4] E. Casey, S. Barnum, R. Griffith, J. Snyder, H. van Beek, A. Nelson, Advancing coordinated cyber-investigations and tool interoperability using a community developed specification language, Journal of Digital Investigation 22 (2017) 14–45. URL: https://doi.org/10.1016/j.diin.2017.08.002.

[5] M. Pollitt, E. Casey, D.-O. Jaquet-Chiffelle, P. Gladyshev, A Framework for Harmonizing Forensic Science Practices and Digital/Multimedia Evidence, Technical Report, The Organization of Scientific Area Committees for Forensic Science, 2018. URL: https://dx.doi.org/10.29325/OSAC.TS.0002.

[6] D.-O. Jaquet-Chiffelle, Trust and identification in the light of virtual persons, 2009. URL: http://www.fidis.net/fileadmin/fidis/deliverables/new_deliverables/fidis- accessed: 2018-11-15.

[7] B. Carrier, File System Forensic Analysis, Addison-Wesley, Upper Saddle River, NJ, USA, 2005.

[8] A. Tanenbaum, Modern Operating Systems, 3rd edition ed., Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2008.

[9] E. Casey, Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet, third ed., Academic Press, Waltham, MA, USA, 2011.

[10] E. Casey, Clearly conveying digital forensic results, Digital Investigation 24 (2018) 1 – 3. URL: http://www.sciencedirect.com/science/article/pii/S1742287618301154. doi:https://doi.org/10.1016/j.diin.2018.03.001.

[11] E. Casey, Error, uncertainty and loss in digital evidence, International Journal of Digital Evidence 1 (2002) 2.

[12] ENSFI, Enfsi guideline for evaluative reporting in forensic science, 2015. URL: https://www.unil.ch/esc/files/live/sites/esc/files/Fichiers 2015/ENFSI Guideline Evaluative Reporting, accessed: 2019-04-05.

[13] X-Ways Software Technology AG, X-ways forensics and winhex manual, 2018. URL: http://www.x-ways.net/winhex/manual.pdf, accesses: 2018-11-15.

[14] E. Casey, Digital stratigraphy: Contextual analysis of file system traces in forensic science, Journal of Forensic Sciences 63 (2018) 1383–1391. URL: https://dx.doi.org/10.1111/1556-4029.13722.

[15] D. Farmer, W. Venema, Forensic Discovery, 1st ed., Addison-Wesley Professional, 2009.

[16] E. Casey, R. Zoun, Design tradeoffs for developing fragmented video carving tools, Digital Investigation 11 (2014) S30 – S39. URL: http://www.sciencedirect.com/science/article/pii/S174228761400053X. doi:https://doi.org/10.1016/j.diin.2014.05.010, fourteenth Annual DFRWS Conference.

[17] R. Pittman, D. Shaver, Windows forensic analysis, in: E. Casey (Ed.), Handbook of Digital Forensics and Investigation, Academic Press, Waltham, MA, USA, 2010.

[18] B. Carrier, Test 7 NTFS undelete image #1, 2004. URL: http://dftt.sourceforge.net/test7/, accessed: 2018-11-14.

[19] D. Hitz, J. Lau, M. Malcolm, File system design for an NFS file server appliance, in: Proceedings of the Winter 1994 USENIX Technical Conference, San Francisco, CA, USA, 1994, pp. 235–246.

[20] J.-N. Hilgert, M. Lambertz, D. Plohmann, Extending The Sleuth Kit and its underlying model for pooled storage file system forensic analysis, in: Proceedings of the Seventeenth Annual DFRWS USA, Austin, TX, USA, 2017, pp. S76–S85. URL: https://doi.org/10.1016/j.diin.2017.06.003.

[21] The FreeBSD Project, ZFS tuning guide, 2018. URL: https://wiki.freebsd.org/ZFSTuningGuide, accessed: 2018-11-14.

[22] L. K. O. Inc, Glossary, 2018. URL: https://btrfs.wiki.kernel.org/index.php/Glossary, accessed: 2018-11-14.

[23] A. Caithness, The forensic implications of SQLite's Write Ahead Log, 2012. URL: https://digitalinvestigation.wordpress.com/2012/05/04/the-forensic-implic accessed: 2018-11-14.

[24] P. Sanderson, R. Hipp, B. Shavers, H. Mahalik, E. Zimmerman, SQLite Forensics, Independently published, 2018.

[25] A. J. Nelson, E. Q. Steggall, D. D. E. Long, Cooperative mode: Comparative storage metadata verification applied to the Xbox 360, in: Proceedings of the Fourteenth Annual DFRWS, Denver, CO, USA, 2014, pp. S46–S56. URL: https://doi.org/10.1016/j.diin.2014.05.004.

[26] S. Garfinkel, A. Nelson, D. White, V. Roussev, Using purpose-built functions and block hashes to enable small block and sub-file forensics, Digital Investigation 7 (2010) S13 – S23. URL: http://www.sciencedirect.com/science/article/pii/S1742287610000307.

doi:https://doi.org/10.1016/j.diin.2010.05.003, the Proceedings of the Tenth Annual DFRWS Conference.