On Metadata Context in Database Forensics

Martin S Olivier

ICSA Research Group, Computer Science, University of Pretoria, South Africa

Abstract

Database Forensics is an important topic that has received hardly any research attention. This paper starts from the premise that this lack of research is due to the inherent complexity of databases that is not fully understood in a forensic context yet. The paper considers the relevant differences between file systems and databases and then transfers concepts of File System Forensics to Database Forensics. It is found that databases are inherently multidimensional from a forensic perspective. A notation is introduced to express the meaning of various possible forensic queries within this multidimensional context. It is posited that this notation, with the multidimensional nature of databases as described, forms a map for possible Database Forensics research projects.

Key words: Database forensics, metadata, relational databases, database, integrity, forensic notation

1 Introduction

The advantages of databases over files are well known. Many of those advantages stem from the fact that databases also contain metadata about the data it stores. Such metadata often describes the data so that programs that use the database are independent of the physical representation of the data [1]. Furthermore, such metadata is used to link individual records to one another. In a nutshell, metadata imbues meaning to a database that is not present in the raw data itself.

Clearly databases often contain information that may be useful during many forensic investigations. This will be the case where a victim's database may contain information useful to the investigation, but also where a suspect has used a database to facilitate his or her suspicious acts.

Email address: `http://mo.co.za` (Martin S Olivier).

The fact that output produced by a database is a function of both the raw data it contains, as well as the metadata it contains, has significant forensic ramifications that — as far as we are aware — have not yet been explored in the scientific literature at all. To illustrate just some of these ramifications, consider the following examples. A cracker breaks into a shop's database, and interchanges the cost price and selling price columns (assuming a relational database). This leads to a significant loss for the shop, despite the fact that the 'real' data has not been modified in any way. As a second example, suppose a criminal wants to hide the existence of certain data; this criminal stores the data in a column in a relational database using a column name (attribute) that does not indicate its purpose. In fact, the data may be stored in several tables, using a series of foreign keys to link them. A view may enable easy access for the criminal, but the moment the view — stored as metadata — is deleted, the logical relationship between these items is lost. Other examples readily come to mind: Deleted stored queries may have contained the essence of a criminal's use of a database. Entire tables may disappear from view the moment their definitions are removed (even though the raw data may still exist). And so the list continues.

To some extent the problem posed to forensic examiners of a database is similar to that posed by a so called *live* investigation of an operating system [2]: In the case of the latter, the data ostensibly retrieved from the system may not be what is really on the system, but what the rootkit (or other malware) prefers to present. In the case of a database, the retrieved information is similarly coloured by the metadata. However, in the case of a database system, a *dead* (or offline) analysis does not solve the problem: it is currently likely that the same metadata will be used to interpret the raw data retrieved from the database.

In fact, database rootkits have begun to receive some attention in online and other forums in the recent past — typically in the context of a rootkit for a specific product, such as 'Oracle rootkits'. However, very little has been published on the topic in archival sources; we have found one book [3] that discusses the topic.

Despite the potential value of databases and their popularity, and despite the pitfalls given the role that metadata plays, Database Forensics has received surprisingly little attention in the scientific literature. In fact, it seems that the term *database forensics* has not yet been explored beyond its intuitive meaning, namely that it deals with forensics of databases. The work that has indeed been done in this context, and — more specifically — the *lack* of work done in this context, is explored more fully in Section 2 below.

Given this lack of scientific publications on the topic it is necessary to consider the foundations on which a theory of Database Forensics may be built. The similarities between Database Forensics and File System Forensics are obvious: both focus on the retrieval of stored content (as well as stored metadata), but in different environments. Moreover, the field of File System Forensics is well understood, and often used in practice. Hence, one possible approach is to consider the similarities and differences between these two contexts, and then derive principles for Database Forensics from these similarities and differences. This is exactly what this paper does.

The paper fleshes out the field of Database Forensics to explore possibilities that should be developed during subsequent research in this area. The intention of the paper is to cover the broad topic, rather than specific details.

More specifically, the paper intends to consider databases as multidimensional constructs, compared to one dimensional file systems. Given this multidimensionality, it is necessary to derive a notation that expresses the various meanings a forensic query may intend to express.

The remainder of the paper is structured as follows. Section 2 explores the (lack of) work done in the field of Database Forensics. Section 3 highlights the critical differences between databases and files from a forensic point of view. Section 4 contains a brief overview of File System Forensics. Section 5 then presents our tentative view of the broad field of Database Forensics. Section 6 concludes the paper.

2 Background

This section briefly reviews the little work that the author was able to find on the topic of Database Forensics in the scientific literature. However, its primary intention is to highlight the lack of work done in the field.

The realisation that Database Forensics is an area that deserves attention is not new. The IFIP WG 11.3 Calls for Papers have included *Database Forensics* as a topic of interest since 2003. However, none of the conferences since then [4,5,6,7,8,9] addressed this topic. A panel at the 2003 conference [10] did discuss Digital Forensics, but did not specifically address Database Forensics.

Casey and Friedberg [11] also mention Database Forensics as a topic that deserves attention, albeit in a more practical context than a 'pure' scientific context.

No paper on Database Forensics has been published in the journal *Digital Investigation* since its publication started in 2004 or the *International Journal of Digital Evidence* since publication started in 2002. Neither was Database

Forensics explicitly covered at any of the recent meetings of IFIP WG 11.9 (on Digital Forensics) [12,13,14,15] or that of the Digital Forensics Research Workshop [1] (DFRWS).

In fact, at the time of writing this a search for the (quoted) phrase "Database Forensics" on `scholar.google.com` yielded only twelve hits; most of these twelve hits were irrelevant. The relevant ones were the Casey and Friedberg editorial [11], a student project paper [16] and a US patent [17]. They are touched on again below. The ACM Digital Library [2], IEEEXplore [3], and Microsoft's `academic.live.com` found no matches. ScienceDirect [4] found twelve matches because it ignored the quotation marks in the search; only the Casey and Friedberg editorial [11] was relevant.

The student project paper mentioned above [16] is a short report that finds that some information that has been deleted from MySQL and PostgreSQL is still visible in the lower level files. This finding is explored in more depth in two subsequent papers [18,19], but the emphasis is on the balance between privacy and accountability, rather than on Database Forensics *per se*.

The patent [17] seems to patent some apparently generic steps that a forensic investigator may try to follow to discover more information about an operation that was performed on a database. The term Database Forensics is not explored.

None of the textbooks on forensics available to the author mentions Database Forensics. As one final attempt at this time, `books.google.com` was used to search for the phrase "database forensics". One accidental match was found, while "digital forensics" matched 188 books, and "computer forensics" 716 books. The latter numbers indicate that a fair number of books from the broader field have been included in the Google database; the fact that no match has been found for the specific phrase indicates that a manual search is unlikely to yield results.

Hence, one may safely conclude that the term Database Forensics has not been established in the scientific literature yet.

This does not imply that various investigations based on database forensics have not happened in practice. Arguably the most complete practical resource available is a series of papers by Litchfield [20,21,22,23,24,25]. The papers focus on Oracle databases (more specifically, Oracle 10g Release 2 server running on Windows). The papers address information available from redo logs, dropped

[1] http://www.dfrws.org
[2] http://acm.org/dl
[3] http://ieeexplore.ieee.org
[4] http://www.sciencedirect.com

objects and other sources — as should be clear from their titles. However, they do not consider a generic underlying model to serve as basis for these practical techniques. A book by the same author is expected soon [26].

Another book that considers Oracle forensics in detail is that by Wright [3]. Again the book is written at a practical level and intended for the database administrator, rather than an attempt to focus on the underlying theory. It does, amongst others, consider the impact of rootkits on an investigation in an Oracle environment. An earlier paper by the same author investigated the possibility of using Oracle LogMiner as a forensics tool [27].

Similar to the work on Oracle Forensics, it seems that Microsoft's SQL Server is beginning to attract attention in this regard. One paper that describes a specific incident exists [28]. A book on SQL Server forensics is due soon [29].

Most of the remaining books that mention database forensics seem to do so in passing (and without mentioning Database Forensics explicitly). We conclude with two examples. Anastasi [30, p.204] describes a fictitious database forensics investigation that contains some details that render it realistic. (Some questions have indeed been asked about exactly how fictitious the account given in the book is [31].) Mohay et al [32] highlight that reverse engineering may be used to recover the structure of a database to be investigated.

Much of this paper focuses on the (forensics use of the) layered nature of a Database Management Systems (DBMS). Note that Carrier [33] has previously considered the (generic) impact of abstraction layers on analysis tools.

3 File Systems compared to Databases

In order to identify the critical differences between file systems and databases one needs to consider the salient features of both systems.

3.1 File systems

File systems are relatively simple: On most current systems, the Unix notion of files as streams of bytes has prevailed. These files are organised using a hierarchical approach where directories are files that contain other files. Various attributes (metadata), such as file name and time of creation, are associated with each file. On a lower level, files are represented using a file system, such as FAT, NTFS or *ext3*. Metadata on this lower level is used to map files to their physical positions on disc, as well as to maintain information about free space on the disc. Virtually any textbook on Operating Systems (such as [34]) may be consulted for further details.

From a forensic perspective the following aspects are known to be useful:

- The contents of a file (for example [35]);
- Metadata about a file [36,37];
- Reconstruction of files where low-level meta-data has been lost or damaged (known as *file carving* [38]); and
- Recovery of information that has been stored on free parts of the disc [37]

(It is worth pointing out that Carrier (as cited by Eckstein [37]) divides meta-data in four layers: the file system layer, the data unit layer, the inode layer and the human interface layer; for our purposes, the two layers above suffice.)

For technical details about File System Forensics, see the book by Carrier [39].

Database Forensics is similar to File System Forensics in the sense that the investigator needs to find information in the database, information about that information and/or recover lost or deleted information. Database Forensics differs from File System Forensics since databases have a significantly more complex structure.

3.2 Databases

In order to gain some insight into the forensically interesting aspects of a database we need a comparable abstract picture of a database. The ANSI/SPARC reference model [40,1] is the classic work done in this regard. The reference model [40]

> *"is based on a two-dimensional classification of data. The well-known point-of-view dimension consists of external, conceptual, and internal schemas; and the orthogonal intension-extension dimension consists of the data model, data dictionary, and application schemas, and the application data."*

Since these dimensions are orthogonal, this implies that at least twelve points exist on which a forensic examination may focus. Moreover, such an investigation may have to consider (the lack of) correlation about data from these perspectives.

To illustrate this, consider the following: A forensic investigation may need to know what the data dictionary says about the conceptual layer (or, what the conceptual layer looks like, given the data dictionary). Indeed, the data dictionary may be the target of an attack. The purpose of such an attack may be the destruction of the data dictionary, or subtle changes in the data dictionary that yield subtly different answers to queries than the correct answers. To illustrate how the data dictionary may impact the 'data' on the database, consider two cases. Firstly, integrity rules are typically stored in the data dictionary [40]. In order to get data into the database that does not conform to the integrity rules, these rules in the data dictionary may have to be modified by the attacker. Secondly, the attributes of an entity are described in the data dictionary. Omitting an attribute (or changing its name) effectively deletes that data. Interchanging two attributes (say PURCHASE_PRICE and SELLING_PRICE, as alluded to earlier) may change critical data as presented by the database. The data dictionary also contains information that may be of forensic interest itself, such as the creation time of an entity — whether that entity occurs on the external, conceptual or internal layer.

In fact, the number of points of interest may be more than twelve: The external schema defines the data to be provided to a specific user. Different schemas may yield different views of the data to different users; these different views may be pertinent during a forensic investigation. And the number of such external schemas only depends on the database being considered.

In addition to this, the operating system's management of the files used for the physical layer forms further points of forensic interest.

On the other hand, the number of points of interest may be fewer than expected given the discussion up to this point. The data model layer of the intension-extension dimension, for example, models the data model (such as the relational model) being used in the database. For reasons of efficiency, this layer is typically hardcoded in the DBMS [40]. While it may be attractive to build a forensic tool that is generic in the sense that it will work for any data model, this is impractical for three reasons: Firstly, given the fact that the data model is typically hardcoded in the DBMS implies that such a model is not necessarily available to be used by a forensic tool, and therefore new model representations will have to be created. Whether such a model corresponds exactly to the hardcoded version used by the DBMS is a forensic concern. Secondly, just as efficiency is a concern for a DBMS, it is a concern for forensic tools. Finally, if the vast majority of database systems use a particular data model, the cost of developing a generic tool may not be justified. Therefore it seems far more likely that relational forensic tools will be developed. However, since one of the purposes of this paper is to flesh out the alternatives it cannot rule out the possibility of generic tools.

Note that the purpose of each layer in the intension-extension dimension is twofold: On the one hand it describes its own particular layer (the intension); on the other it defines the schema for the next higher layer [40]. For example, the data dictionary layer contains the data dictionary 'data' as well as the schema of the application schema layer. The implications this holds for a forensic tool will be discussed in Section 5 below.

Some may argue that the ANSI/SPARC reference model is overly idealistic (or even dated). Other options were indeed also available to use as basis for the exploration of Database Forensics. The original ANSI/SPARC architecture [41] specified 42 interfaces between various components. Such interfaces may be used as the basis to explore Database Forensics. Similarly, one may prefer to use a component-based architecture or a detailed functional decomposition of a DBMS as the basis. This paper chose the ANSI/SPARC Reference Model because it seemed to be the most fruitful.

One final respect in which databases differ significantly from most file systems is the level of logging that occurs in a database. Such logging may include enough information to roll a database back or forward. Logging will play a significant role when Database Forensics is considered in Section 5 below.

4 File System Forensics

The forensics process is typically divided into phases, such as preparation, acquisition, analysis and reporting. For the purposes of this section we do not want to consider these phases in detail, but extract those parts from it that may be considered as File System Forensics.

Those parts of the investigation that form part of File System Forensics typically proceed as follows. Firstly the computer containing the data is located and (usually) immediately switched off to avoid any further changes to data. The disc is then unplugged and a copy of the disc is made during a process known as *imaging*. Various steps are taken to ensure that imaging occurs in a forensically sound manner. When possible, physical write blockers are used to prevent any write operation to the evidence disc. An MD5 (and/or other digests) is calculated for the disc as well as for the image to ensure (and prove) that the image corresponds to the original contents.

Next the image is analysed. This involves searching through the image for specific contents or the content is inspected using various tools to find possible evidence. It is known that persons who engage in illegal activities often take steps to hide evidence of their activities. In the case of file systems, portions of the disc that are usually ignored by the operating system are often used as a hiding space for information related to such activities. So called slack space — the space between the end of logical files and the end of the last allocated sector or cluster — is one example. Partitions that do not use a known file system format is another example. Furthermore, files containing evidence may have been deleted and the analysis should therefore also consider deleted space.

In some cases the file structure may have been damaged (or purposively stored on a separate medium). In such cases the investigator is faced with the arduous task of reassembling files that may be spread over the disc. As indicated earlier, the process of reassembling such files (and separating them from the remainder of the contents on the disc) is known as *file carving*.

The process, as described above, is often not as simple or clear cut. For example, the question whether the systems should indeed be switched off immediately is the subject of considerable debate [2,42]. Analysing a system without switching it off is known as *live* analysis, whereas analysis after it has been turned off is known as *dead* analysis.

Despite such differences, three fundamental elements may be abstracted from the various approaches:

Integrity: In order so be forensically sound the copy should be a provably exact copy of the original data (or, if the original data is analysed, it should be clear that the data has not been tampered with). This requirement is seen as more critical than in other branches of forensics due to the ephemeral nature of data, as well as one's inability to distinguish between data stored before seizure and modifications made after seizure.

Searchability: Analysis primarily consists of locating evidence in the typically large amount of information that has been seized. This is typically achieved through browsing, text searching, or a more sophisticated form of searching. Correlation of various sources of data forms part of this element.

Restoration: It is often important to isolate information that has already been deleted, that always existed outside the normal file structures (in slack space or elsewhere) or where the physical organisation has been lost (using file carving).

Metadata extraction: Files are often associated with useful metadata, including the date and time of creation, modification and last access.

While not usually seen as part of File System Forensics, a fifth element may be added to this list. *Attribution* refers to the question of *who* caused a certain effect in a file (or in other evidence) [43,44]. While certainly debateable, we argue that some aspects of attribution may be categorised as part of File System Forensics.

Attribution: In the File System Forensics context attribution may be based on the style of the contents of a file, or from metadata, such as file ownership and/or authorisation.

In the next section these five elements will be combined with the characteristics of a DBMS that were derived in Section 3 to provide a possible map for

5 Database Forensics

Given the two-dimensional nature of a DBMS highlighted in Section 3 above, it seems logical that a forensic investigation of a database will need to consider multiple dimensions. Which dimensions play a role in the investigation may depend on the nature of the incident being investigated. However, ignoring a dimension may present an opportunity for a guilty party to successfully challenge evidence that could have lead to his or her conviction.

The relevant dimensions are:

(1) The point-of-view (internal, conceptual and external schema) dimension — from the original ANSI/SPARC work [41];
(2) The intension-extension dimension — from the ANSI/SPARC Reference Model [40]; and
(3) The version dimension, given the ability to roll the database back or forward, or, at the very least, an ability to interpret logs.

To illustrate the role of these dimensions consider queries such as the following, all of which are conceivably useful for some forensic investigation:

- What are the current data values that would be displayed if user u_1 requests some specific records?
- What were the data values (conceptually) contained in the database for some records at time t_1?
- What schema would have been used to answer query q_1 when user u_2 issued it?
- In all of these cases, who wrote that particular value (or schema) to the database?

The ability to answer questions about what value was stored in a database field at some specific point in the past may be idealistic — especially for fast changing data. However, for schemas and similar metadata that change relatively infrequently, this may often be a realistic question.

Before we return to such possible forensic queries let us remember the elements of a forensic investigation in file systems as described in the previous section: integrity, searchability, restoration, metadata extraction and attribution.

From a *searchability* perspective it seems that the best forensic tool is the database itself. The database, after all, enables the investigator to search data using powerful queries. When necessary, it is also possible to easily browse the database contents. In order to formalise matters, we will use relational algebra; for example to select some record from a TABLE with $id = 1$ we will write

$$\sigma_{id=1}(\text{TABLE})$$

However, using the database one is immediately faced with questions about *integrity*. These issues are similar to those encountered in File System Forensics, but more varied. In the case of File System Forensics two questions are levied: Is the copy an exact copy (as evidenced by MD5 digests) and/or can I trust the software reading the disc to give a faithful indication of what is on the disc? In the case of Database Forensics the question about an exact copy may still be asked. However, even if one uses an exact copy with a trusted database executable, the executable may still be coerced into giving false results if, say, the data dictionary has been modified. This leaves one with the question: should one use the data dictionary as it occurs on the system being investigated, since that gives the best indication of what users would see? Or should one try to get a known good dictionary to see the 'actual' database contents?

Consider the intension-extension dimension more closely. Given the fact that the data model is often hardcoded as part of the DBMS, the question of whether the data model should be used for the investigation as it occurs on the system, or whether a 'clean' (or 'known good') data model should be used, is similar to the question in File System Forensics whether a live analysis (with a possibly compromised operating system) should be performed, or whether a dead analysis with known good utilities is to be performed. The implications of making this choice are similar to the implications the choice has in File System Forensics.

To be precise, it is therefore necessary to specify which data model (actually which DBMS) has been used. Assume that dm is the name (and/or version) of the data model. Assume further that we will decorate it with an asterisk (as dm^*) if it is a known clean version of the data model. It will be decorated with a subscript if it is some identified earlier version (which may, or may not be known to be good). So dm_5 and dm_5^* are both options as long as it is known which is 'version 5' of the data model. The dm_5 version is the version that existed on the DBMS at that time; the dm_5^* version is the one that should have existed on the DBMS at that time. Of course, it is entirely possible that

$dm_5 = dm_5^*$. If no subscript is used it indicates the version current at the time of the investigation. Hence

$$\sigma^{dm}(\text{TABLE})$$

corresponds to a live analysis in File System Forensics, while

$$\sigma^{dm^*}(\text{TABLE})$$

corresponds to a dead analysis.

However, as already alluded to, a dead analysis does not *per se* guarantee the integrity it did in File System Forensics: a compromised data dictionary will affect the results of even a clean DBMS. Hence we need to introduce similar notation for the data dictionary than was used for the data model. Therefore dd_n^* refers to version n of the data dictionary, and the asterisk indicates that it is a known good version. Both the asterisk and the subscript may be omitted as was the case for dm.

Similar arguments apply for the application schema (as) and the application data (ad). A fully specified query used during an examination may therefore look as follows:

$$\sigma_{query}^{dm^*;dd^*;as^*;ad}(\text{TABLE})$$

As specified, this query is to be performed using a known good data model, data dictionary and application schema, but the data as found on the disc.

In fact, this example suggests a general approach to conduct a typical analysis. First, decide on whether a live or dead analysis is to be performed. If a live analysis is to be performed it is necessary to show that $dm^* = dm$. Next, show that $dd = dd^*$. Then show that $as = as^*$. Finally, conduct the analysis using dm or dm^*, with dd and as, which may now both be decorated with an asterisk. This implies that the evidence cannot be successfully challenged with a claim that it is not trustworthy because, say, the data dictionary was compromised.

Obviously, integrity requires that dd (or as or ad) indicates *exactly* what occurs on the disc(s). This may mean that an image of the disc has been made and the MD5s verified, or that some live audit system that will be trusted by the courts is in place. In the database context neither of these options is simple. However, the current paper does not consider them further.

Up to this point we have assumed, for the sake of convenience, that queries will yield the same result irrespective of who issues them. It is, however, clear from the remainder of the paper that one needs in many cases to explicitly consider the external schema exported to given users. This will be indicated by specifying the user from whose perspective a query needs to be considered as part of the query. This user will be specified after the table(s) on which the query is to be performed, as follows:

$$\sigma^{dm^*;dd^*;as^*;ad}_{query|u_1}(\text{TABLE})$$

In this example it has been specified that the point of view of user u_1 should be used for the query.

The conceptual and external schema layers of the point-of-view dimension have now been addressed. This leaves (at least) the internal schema layer. Exactly how queries (or analysis) may be performed at this layer is not clear. However, consistency suggests that a query at this layer may be expressed as

$$\sigma^{dm^*;dd^*;as^*;ad}_{query|internal}(\text{TABLE})$$

Since the purpose of the current paper is to raise such possibilities, rather than explore them in depth, assigning a suitable meaning to this construct is left for future research.

While the lowest layer of the point-of-view dimension is the internal schema, the ANSI/SPARC work does consider the interface with the operating system. Since File System Forensics is relatively well developed and often deals with matters that have no relationship to databases at all (such as slack space utilisation) it seems logical to see the operating system layer as the lowest layer of the point-of-view dimension of Database Forensics. In a similar vein to the construct formulated for the internal schema, it seems logical to introduce an operating system layer query, as follows:

$$\sigma^{dm^*;dd^*;as^*;ad}_{query|os}(\text{TABLE})$$

Again, the possible meanings of this construct are not obvious and finding an appropriate meaning (if it exists) is left for future research.

Queries for forensic purposes have, up to this point in the paper, tacitly assumed that application data is to be queried. It is clear that in some cases one may want to consider, say, the data dictionary layer. It is assumed that all data structures are represented using the data model of the database. Hence queries against the data dictionary may be issued in exactly the same manner as queries against the application data. Two differences need to be considered. Firstly, the query will obviously identify a table from the data dictionary, and not one that contains application data. Secondly, the state of the higher levels of the intension-extension dimension will be immaterial when a lower level is queried — and therefore may be omitted from the query specification. To illustrate, suppose a query is to be used to obtain data from some table DDTABLE in the data dictionary. This query may look as follows:

$$\sigma^{dm^*;dd}_{query}(\text{DDTABLE})$$

Selection is clearly not the only operation that is useful to search the database for evidence. Projection and joining of tables are also important. Both these operations depend on the lower levels of the intension-extension dimension and therefore need to be decorated with the same superscripts and subscripts used for selection.

It is an open question whether it will be forensically useful to select two sets of results using a given combination of data model, data dictionary, application schema and application data, and then to join them in a different environment.

Before leaving the current thread, it seems useful to introduce two further notational conventions. Firstly, thus far the conceptual schema has been identified by omitting it from a query specification. In some circumstances, it may be useful to indicate it explicitly to avoid ambiguity, as follows:

$$\sigma^{dm^*;dd^*;as^*;ad}_{query|conceptual}(\text{TABLE})$$

Finally, it is of forensic use to indicate that the versions of the data model, data dictionary, application schema and application data as they existed at some point in time is to be used. We indicate that these versions are implied as they were at time t_1 by writing:

$$\sigma^{dm^*;dd^*;as^*;ad@t_1}_{query|u_1}(\text{TABLE})$$

The @ specification applies to all components listed in the superscript. If one wants to use only, say, the application data as it appeared at time t_1 it is suggested that one should find the appropriate version, say ap_i, and specify it in the query. In some investigations it may be of particular importance to note that some data has been deleted, and present evidence that such data did exist in the database at some earlier time.

From File System Forensics the paper highlighted integrity, searchability, restoration, metadata extraction and attribution. The discussion up to this point considered searchability and integrity. In a database, metadata occurs in the lower layers of the intension-extension dimension, and has thus been dealt with. This leaves us with the questions of restoration and attribution.

5.2 Restoration

In this paper the term *restoration* is used to refer to the recreation of data that has been (partially) destroyed, or only partially recovered by a forensic capture process. Remember that, in file systems, this process is known as file carving. In a database, many forms of carving are conceivable. Record carving, for example, may refer to the recovery of records from some medium. Given the regular structure of records, this will, in many cases, be simpler than general file carving. Table carving may, similarly, refer to the recreation of tables from carved records. However, not all records with a similar structure necessarily formed part of the same table; other hints — such as links in other tables — will have to be used to confirm whether records do or do not belong to the same table. Just as application data may be carved, is it possible to carve data from the lower layers of the intension-extension dimension. In addition to carving, it may be possible to reconstruct data on such lower layers: given the regular structure of a carved table, one may derive the schema for it; however, some parts of such a reconstructed schema — such as element names — may have to be guessed. Also, not all metadata — such as time of creation — will be useful after such a construction. Hence it is necessary to distinguish between restored data that has been restored by carving, and restored data that has been restored by reconstruction. In this paper a single prime ($'$) will be used to indicate carved data, while a double prime ($''$) will be used to indicate reconstructed data. A query specification that states

$$\sigma^{dm^*;dd^*;as'';ad'}_{query|conceptual}(\mathrm{TABLE})$$

therefore should be interpreted as follows. The *query* has been executed on a known good data model. A known good data dictionary was found (perhaps from the design or backups, to be clearly documented.) The application data was carved from one or more discs. The application schema was, however, reconstructed.

A detailed consideration of forensic recovery in databases falls outside the scope of the current paper, and is left for future research. However, it is worth pointing out that much of the (practical) work mentioned in Section 2 deals with carving application data (ad') and inferring (or reconstructing) application schemas (as''). Given the fact that such work often have to recover data from proprietary formats, it is often necessary to reverse engineer not only the application schema and other data, but also the underlying DBMS structure of the (known) DBMS used to manage the data.[5] This not only makes the work particularly challenging, but may also be the basis of sufficient doubt to

[5] The author wishes to thank the anonymous referees for suggesting many of the ideas that follow in the remainder of this section.

challenge such evidence in court. The seriousness of this problem clearly depends on what needs to be proven in a specific case — in some cases the mere existence of a character string on a disc may be sufficient proof, while proof in other cases may depend on the semantics of data in a partially destroyed database. When one considers the inherent complexity of the data structures used, as well as the fact that the actual data structure may depend on the version, installation parameters, operating system and other factors, the magnitude of the potential reverse engineering effort becomes evident. Moreover, many DBMS licences prohibit reverse engineering. Clearly vendor support is required to adequately address this problem.

Reconstruction is not limited to the higher layers (ad and as): In some cases the exact version of the DBMS used (and hence, the appropriate dm) may not be clear and one has to infer which version had been used. In general, fingerprinting is required that ideally should be able to determine the version with an acceptable level of certainty. In fact, such fingerprinting should be possible to not only determine the version, but also the product (or product family) used. In some cases one may form a hypothesis based on which DBMS is (or DBMSs are) installed on the computer to be analysed — and then confirm the hypothesis by verifying that the recovered data has the expected structure. In other cases, only recovered data will be available and the DBMS will have to be inferred from the data.

It is, of course, possible that one may encounter data that exhibits the regularity associated with databases, but where the product cannot be identified. The question then is whether it will (sometimes) be possible to infer the data model with enough certainty to be useful. An example may illustrate this: it is simple to modify an open-source DBMS so that recovered data may not correspond with the known versions of that DBMS, but may still exhibit many similarities with (versions of) that DBMS. The open question is with what level of confidence an expert witness will be ably to testify in court that the recovered data has been interpreted correctly. Stated in the terms of the paper: with what certainty can statements about dm'' and dd'' be made if they are not recognised as specific products?

5.3 Attribution

The final issue to be addressed is that of attribution.

Attribution in Database Forensics may be approached from two angles. Firstly, if detailed logs are available, such logs may be used for attribution. However, it should be kept in mind that the perpetrator may have been able to modify the logs or bypass the logging facility. Secondly, the metadata may be used to determine who was authorised to perform a certain action and use that as the basis for attribution. In this case it should be borne in mind that the perpetrator may have bypassed the authorisation system and hence not be found using this approach.

A combination of these two approaches will also be useful since they will enable one to correlate findings.

A further consideration is whether the user implicated by either or both of the mechanisms mentioned above did indeed commit the act. For example, it is possible that someone else stole this person's password and misused it to perform the action being investigated. However, we contend that determining the identity of the user is as far as Database Forensics can take the matter. Beyond that, attribution becomes an issue for other branches of forensics, including but not limited to the broader branch of Digital Forensics.

There is no reason why both of these sources for attribution (the logs and the authorisation information) cannot occur in the database (and often they do). Therefore, suitable phrased queries in the format described above will already be able to answer such questions. However, the precise formulation of the query depends on the DBMS being used. Given the centrality of the issue of attribution it is desirable to express one's intentions more clearly. This may be done as

$$\omega_{\text{log}}^{dm^*;dd^*;as^*;ad}(a_i)$$

for attribution based on the logs, or

$$\omega_{\text{auth}}^{dm^*;dd^*;as^*;ad}(a_i)$$

for attribution based on authorisation rights. Here a_i is some action that was purportedly performed at some specified time. Actions are seen to be the execution of typical SQL statements or queries expressed in relational algebra.

It is not clear that the given notation is appropriate. For example, are all the superscripts used necessary? The current paper does not consider this issue further. In the context of this paper it was only important to introduce it for three reasons:

(1) Attribution may be taken significantly further in Database Forensics than in File System Forensics and hence will probably form an important part of Database Forensics;
(2) It illustrates that some forensic questions are important enough to justify their own questions (and notation). It is entirely possible that a range of such issues exist.

(3) Even as preliminarily formulated it poses interesting questions: How is this issue addressed in current DBMSs? To what extent can it be answered using (just) SQL?

6 Conclusion

The intention of this paper was to flesh out issues of Database Forensics with the hope of stimulating research in this important area. It approached its task by identifying various dimensions of Database Forensics: The point-of-view dimension, the intension-extension dimension and the version dimension. In addition it abstracted the elements of an investigation in a File System Forensics context and considered them in a Database Forensics context. A concise notation was introduced to enable one to refer to specific contexts during a forensics investigation with a reasonable degree of clarity. It was argued that some issues (specifically attribution) are important enough to justify their own notation.

In its attempt to cover the field as broadly as possible, the paper did not delve into details. A specific omission was the detailed discussion of what specific queries might look like during specific forensic examinations. What queries does one expect during a child pornography case? What queries does one expect during a financial fraud case? What queries does one expect during a cracking case? Indeed, can general queries be given for such broad categories of cases? Some categories seem promising, but perhaps not all.

It seems worthwhile to consider each of the intersections of points in the various dimensions. Some of them are bound to lead to interesting questions that have not yet been explored, but that will become important during some future real-world case.

When the version dimension was introduced it was suggested that it will enable one to ask questions about the data returned by the database at some point in the past. To what extent can current systems already achieve this goal? What needs to be added to make it work as well as possible?

The current state of the art is to image discs and work from the images, but this is often no longer practical [42] — particularly in the case of databases. Distributed databases only exacerbate this problem. The paper suggested that some alternative solution that will be accepted in court be found — specifically for database environments.

How hard is it to carve records, tables, schemas and other data from damaged database files?

In a nutshell: this paper is intended to raise questions, rather than to answer them. Hopefully it will stimulate readers and some of the answers to these questions will turn out to be interesting.

References

[1] C. J. Date, An Introduction to Database Systems, 8th Edition, Addison Wesley, 2004.

[2] B. D. Carrier, Risks of live digital forensic analysis, Communications of the ACM 49 (2) (2006) 56–61. doi:http://doi.acm.org/10.1145/1113034.1113069.

[3] P. M. Wright, Oracle Forensics — Oracle Security Best Practices, Rampant Techpress, 2008.

[4] S. De Capitani di Vimercati, I. Ray, I. Ray (Eds.), Data and Applications Security XVII: Status and Prospects, Kluwer, 2004.

[5] C. Farkas, P. Samarati (Eds.), Research Directions in Data and Applications Security, Kluwer, 2004.

[6] S. Jajodia, D. Wijesekera (Eds.), Data and Applications Security XIX, Vol. 3654 of LNCS, Springer, 2005.

[7] E. Damiani, P. Liu (Eds.), Data and Applications Security XIX, Vol. 4127 of LNCS, Springer, 2006.

[8] S. Barker, G.-J. Ahn (Eds.), Data and Applications Security XXI, Vol. 4602 of LNCS, Springer, 2007.

[9] V. Atluri (Ed.), Data and Applications Security XXII, Vol. 5094 of LNCS, Springer, 2008.

[10] M. Pollitt, M. Caloyannides, J. Novotny, S. Shenoi, Digital forensics: Operational, legal and research issues, in: S. De Capitani di Vimercati, I. Ray, I. Ray (Eds.), Data and Applications Security XVII: Status and Prospects, Kluwer, 2004, pp. 393–403.

[11] E. Casey, S. Friedberg, Moving forward in a changing landscape, Digital Investigation 3 (1) (2006) 1–2.

[12] M. Pollitt, S. Shenoi (Eds.), Advances in Digital Forensics, Vol. 194 of IFIP, Springer, 2005.

[13] M. S. Olivier, S. Shenoi, Advances in Digital Forensics II, Vol. 222 of IFIP, Springer, 2006.

[14] P. Craiger, S. Shenoi (Eds.), Advances in Digital Forensics III, Vol. 242 of IFIP, Springer, 2007.

[15] I. Ray, S. Shenoi (Eds.), Advances in Digital Forensics IV, Vol. 285 of IFIP, Springer, 2007.

[16] P. Stahlberg, Forensic analysis of database systems — final report, Fall seminar report, Department of Computer Science, UMass Amherst (December 2005).

[17] K. B. Wong, Daniel Manhung; Edwards, System and method for investigating a data operation performed on a database, US Patent (December 2005). URL http://www.freepatentsonline.com/20050289187.html

[18] P. Stahlberg, G. Miklau, B. N. Levine, Threats to Privacy in the Forensic Analysis of Database Systems, in: Proc. ACM Intl Conf. on Management of Data (SIGMOD), 2007, pp. 91–102. URL http://www.cs.umass.edu/~miklau/pubs/sigmod2007LMS/stahlberg07forensicDB.pdf

[19] G. Miklau, B. N. Levine, P. Stahlberg, Securing History: Privacy and Accountability in Database Systems, in: Biennial ACM/VLDB Conference on Innovative Data Systems Research (CIDR), 2007, pp. 387–396. URL http://prisms.cs.umass.edu/brian/pubs/mikalu.cidr2007.pdf

[20] D. Litchfield, Oracle forensics part 1: Dissecting the redo logs, NGSSoftware Insight Security Research (NISR) Publication, Next Generation Security Software, (Version dated 21 March 2007; available at http://http://www.davidlitchfield.com/) (2007).

[21] D. Litchfield, Oracle forensics part 2: Locating dropped objects, NGSSoftware Insight Security Research (NISR) Publication, Next Generation Security Software, (Version dated 24 March 2007; available at http://www.davidlitchfield.com/) (2007).

[22] D. Litchfield, Oracle forensics part 3: Isolating evidence of attacks against the authentication mechanism, Ngssoftware insight security research (nisr) publication, Next Generation Security Software, (Version dated 27 March 2007; available at http://www.davidlitchfield.com/) (2007).

[23] D. Litchfield, Oracle forensics part 4: Live response, NGSSoftware Insight Security Research (NISR) Publication, Next Generation Security Software, (Version dated 20 April 2007) (2007).

[24] D. Litchfield, Oracle forensics part 5: Finding evidence of data theft in the absence of auditing, NGSSoftware Insight Security Research (NISR) Publication, Next Generation Security Software, (Version dated 10 August 2007; available at http://http://www.davidlitchfield.com/) (2007).

[25] D. Litchfield, Oracle forensics part 6: Examining undo segments, flashback and the oracle recycle bin, NGSSoftware Insight Security Research (NISR) Publication, Next Generation Security Software, (Version dated 16 August 2007) (2007).

[26] D. Litchfield, Oracle Forensics Using Quisix, Wiley, 2008, to be published; citation based on information from online retailers.

[27] P. M. Wright, Oracle database forensics using LogMiner, Paper, Next Generation Security Software, (Version dated 10 January 2005) (2005).

[28] K. Fowler, A real world scenario of a SQL Server 2005 database forensics investigation, Information security reading room paper, SANS Institute (2007).

[29] K. Fowler, SQL Server Forensic Analysis, Addison-Wesley, 2009, to be published; citation based on information from online retailers.

[30] J. Anastasi, The New Forensics: Investigating Corporate Fraud and the Theft of Intellectual Property, Wiley, 2003.

[31] N. Lim, J. Anastasi, The new forensics: investigating corporate fraud and the theft of intellectual property, John Wiley & Sons, New Jersey, 2003 (Review), Digital Investigation 3 (4) (2006) 245–246.

[32] G. Mohay, A. Anderson, B. Collie, O. de Vel, R. McKemmish, Computer and Intrusion Forensics, Artech House, 2003.

[33] B. D. Carrier, Defining digital forensic examination and analysis tools using abstraction layers, International Journal of Digital Evidence 1 (4).

[34] W. Stallings, Operating Systems: Internals and Design Principles, 5th Edition, Pearson Prentice Hall, 2005.

[35] N. L. Beebe, J. G. Clark, Digital forensic text string searching: Improving information retrieval effectiveness by thematically clustering search results, Digital Investigation 4 (Supplement 1) (2007) 49–54.

[36] F. Buchholz, E. Spafford, On the role of file system metadata in digital forensics, Digital Investigation 1 (4) (2004) 298–309.

[37] K. Eckstein, Forensics for advanced UNIX file systems, in: Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, IEEE, 2004, pp. 377–385.

[38] S. L. Garfinkel, Carving contiguous and fragmented files with fast object validation, Digital Investigation 4 (Supplement 1) (2007) 2–12.

[39] B. Carrier, File System Forensic Analysis, Addison-Wesley, Boston, Mass, 2005.

[40] Reference model for DBMS standardization, SIGMOD Record 15 (1) (1986) 19–58. doi:http://doi.acm.org/10.1145/16342.16343.

[41] Final report of the ANSI/X3/SPARC DBS-SG relational database task group, SIGMOD Record 12 (4) (1982) 1–62. doi:http://doi.acm.org/10.1145/984555.1108830.

[42] F. Adelstein, Live forensics: diagnosing your system without killing it first, Communications of the ACM 49 (2) (2006) 63–66. doi:http://doi.acm.org/10.1145/1113034.1113070.

[43] P. Juola, Future trends in authorship attribution, in: P. Craiger, S. Shenoi (Eds.), Advances in Digital Forensics III, Springer, 2007, pp. 119–132.

[44] C. Chaski, The keyboard dilemma and authorship identification, in: P. Craiger, S. Shenoi (Eds.), Advances in Digital Forensics III, Springer, 2007, pp. 133–146.

MS Olivier, “On metadata context in Database Forensics” *Digital Investigation*, **5**, 3–4, pp 115–123, March 2009

This is a pre-publication version; published version is definitive.