Chapter 16

SMARTPHONES AS DISTRIBUTED WITNESSES FOR DIGITAL FORENSICS

Heloise Pieterse and Martin Olivier

Abstract Smartphones have become an integral part of people's lives. Their wide range of capabilities and support of diverse applications result in a wealth of data being stored in smartphone memory. Although tools are available to extract and view the data stored in smartphones, no comprehensive process exists for event reconstruction using the extracted data. Data in smartphones is typically stored in SQLite databases and can, therefore, be easily transformed. To perform event reconstruction, multiple SQLite databases have to be integrated. This paper proposes a novel mobile event reconstruction process that allows for event reconstruction by querying the integrated SQLite databases collected from multiple smartphones. The process can create detailed accounts of the events that took place before, during and after an incident.

Keywords: Smartphones, event reconstruction, distributed databases

1. Introduction

Smartphones have become constant companions for people around the world. Their popularity can be attributed to multiple factors, but the most important is their continuous enhancement in functionality. In addition to telephony, smartphones offer functionality similar to a small computer. They support complete operating systems, allow for the installation of third-party applications and have ubiquitous connectivity and communications capabilities.

The extensive and diverse use of smartphones make them a rich source of evidence. Smartphones inherently sense the environment in which they operate and store trace elements (or more) of what they have sensed. The "environment" may include the digital realm (such as social network ecosystems and location-based messages received via Bluetooth

G. Peterson and S. Shenoi (Eds.): Advances in Digital Forensics X, IFIP AICT 433, pp. 237–251, 2014.

or NFC), the physical realm (such as photographs taken, and videos and sound clips recorded) and events that straddle the digital and physical realms (such as wireless access points and cellular towers seen or accessed). The exact events recorded by a smartphone depend on many factors, including the smartphone settings, sensors and applications.

Since smartphones are ubiquitous, there is a high probability that: (i) one or more smartphones would almost certainly have "witnessed" an event that otherwise may not have left any traces; and (ii) multiple smartphones would have "witnessed" the event, which can increase the reliability of the evidence pertaining to the event. To illustrate the latter point, if hundreds of smartphones have recorded an event, it may be possible to determine the time that the event occurred with much greater precision than if information was available from just one or a few smartphone logs.

A set of smartphones essentially forms a distributed database of independent nodes that contain potential evidence about events sensed by the devices. One option is to amalgamate all this information and "mine" it for evidence. However, a more controlled examination of the smartphone data would provide: (i) a structured approach to test hypotheses; and (ii) greater clarity about the reliability of evidence obtained from the distributed database. The autonomy of each smartphone suggests that the set of smartphones should be treated as a federated database. However, practical considerations (such as the time available to obtain evidence from the smartphones) may require that the distributed information be amalgamated into a single database.

Database integration is a well-known problem with significant subproblems. However, a fortunate circumstance reduces the complexity of this process in the current context: smartphones (almost) exclusively use SQLite databases to store information. In addition, the integration does not entail amalgamating information in the usual sense of the word – the different data sets are to be retained as different data sets so that information seen in a data set may be correlated with information seen (or not seen) in other data sets. While these observations do not render database integration a trivial problem, they do suggest that integration in the forensic sense is feasible within the constraints of a forensic examination; they do not equate to the massive projects typically associated with integration efforts.

The immediate challenges are to show that: (i) the remarks about database integration in a forensic context made above are indeed correct; and (ii) useful queries may be answered by an integrated database. This paper provides preliminary evidence that the suggested approach can help address these challenges. The first challenge is addressed by

providing a simple process model based on database integration theory that shows how integration may be achieved. The second challenge is addressed by providing examples of potential queries that demonstrate the utility of the approach.

The outcome of the process is the reconstruction of events surrounding a particular incident by querying the integrated SQLite databases. Event reconstruction supports investigations by allowing multiple events to be tied together and crucial relationships to be identified based on multiple smartphones serving as distributed witnesses.

2. Reconstruction Techniques

Reconstruction is an important component of the digital forensic examination process, allowing for the establishment of the most probable sequence of events that occurred during a specific time period. Forensic Science Central [9] defines reconstruction as the "process of establishing a sequence of events about the occurrences" during and after an incident. Gladyshev [11] describes reconstruction as the process of "determining the events that happened during an incident." Prasad and Satish [21] view event reconstruction as the examination of evidence to determine why it has certain characteristics; this is achieved by using the collected evidence to reconstruct what has happened in a particular system. Finally, Cohen [5] defines reconstruction as an experimental component that is used to test hypotheses.

There have been several attempts to formalize event reconstruction. Formal frameworks are discussed by Stephenson [26], Gladyshev and Patel [12] and Arnes, *et al.* [2]. Stephenson uses a Petri net approach to reconstruct attacks and identify the root cause of attacks. Gladyshev and Patel use state machines and mathematical modeling to reconstruct events. Arnes, *et al.* model a virtual digital crime scene in order to reconstruct events in a realistic fashion.

Techniques such as attack trees and visual investigative analysis also support event reconstruction. Attack trees [22] provide a mechanism for thinking, building, capturing and making decisions about the security of a system. With the help of possible incident scenarios and hypotheses integral to event reconstruction, attack trees can aid the discovery, documentation and analysis of the incident scenarios. Visual investigative analysis is a charting technique that graphically displays sequences of events and relationships using a network approach [16]. This technique was developed to improve the ability to visualize complex crimes [11].

The works described above consider event reconstruction to be a process that is followed to rebuild the possible sequence of events surround-

ing a particular incident. In the remainder of this paper, event reconstruction will, therefore, refer to the rebuilding of events surrounding a specific incident.

Little attention has been directed at the reconstruction of events using smartphones. The techniques discussed above primarily focus on reconstructing events surrounding computer incidents and, although applicable, do not consider the differences associated with using smartphones for event reconstruction. To accommodate the differences, a new process is proposed that allows for the reconstruction of events using data collected from smartphones.

3. Mobile Event Reconstruction Process

Mobile phone forensics is a constantly evolving area of digital forensics [6]. The new millennium introduced a new generation of mobile phones known as smartphones. These devices, with their ability to execute third-party applications, store large amounts and diverse types of data. Evidence commonly found on smartphones includes, but is not limited to, contacts, call history, calendar entries and messages [6]. Several forensic tools are available for retrieving smartphone data (e.g., Oxygen Forensic Suite [17], AccessData Mobile Phone Examiner Plus (MPE+) [1] and viaExtract from viaForensics [28]).

Most of the data extracted from smartphones using forensic tools is in the structured format of SQLite databases, which forms a distributed database of independent nodes containing potential evidence that was acquired and stored by the smartphones. The mobile event reconstruction process described in this paper draws on database integration theory to combine the multiple SQLite databases into a single, integrated multidatabase. The process, which is summarized in Figure 1, involves four distinct phases: (i) extraction; (ii) homogenization; (iii) schema integration; and (iv) exploration. The first three phases perform the database integration while the last phase, exploration, allows for the examination of the data by executing custom queries against the integrated database. The outcome of the process is the reconstruction of events surrounding a particular incident by visualizing the queried data.

3.1 Phase 1: Extraction

The first phase of the mobile event reconstruction process is the extraction of data from smartphones. In addition to SQLite databases, valuable data resides in backup database files in the form of rollback journals and write-ahead log files. These files store the changes made to a SQLite database to enable the database to be returned to a pre-

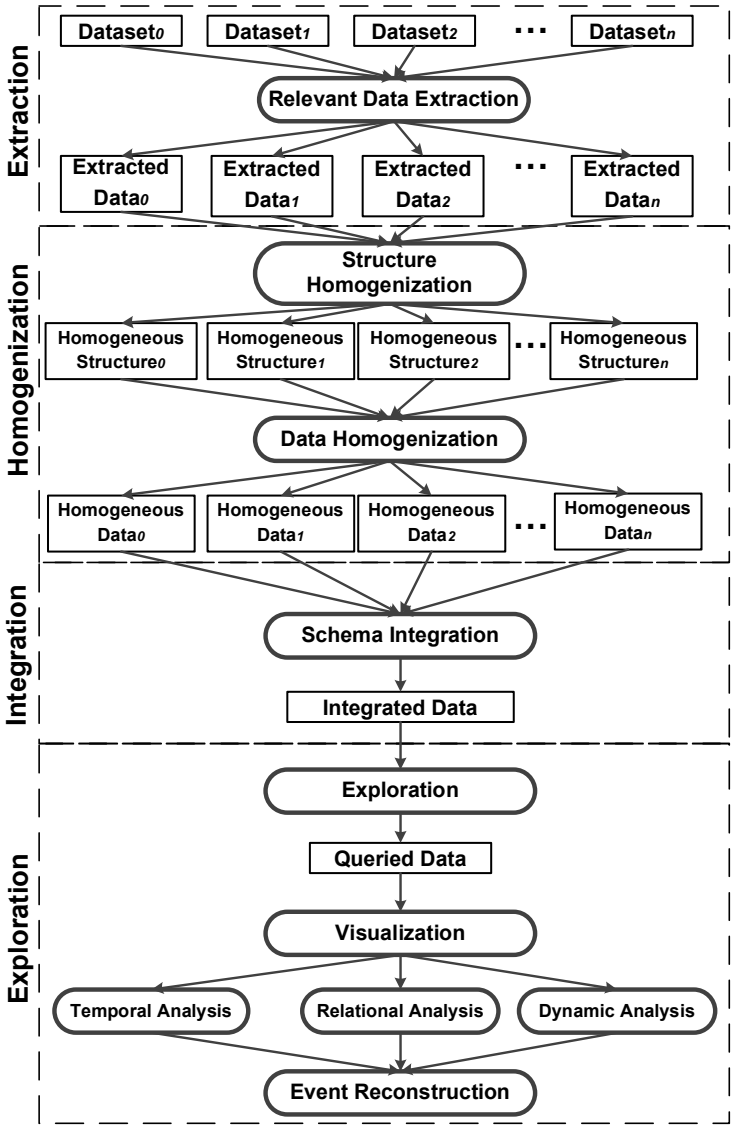

Figure 1. Mobile event reconstruction process.

vious stable state after a crash or failure. Indeed, the files provide an opportunity to extract previously altered or deleted data that can aid in reconstructing events.

- **SQLite Databases:** SQLite is an open source, lightweight database that allows for the quick processing of stored data [14]. Data processing is facilitated by an in-process library that implements

a "self-contained, serverless, zero-configuration, transactional SQL database engine" [23]. Unlike many other SQL databases, SQLite does not require a separate server process and is capable of reading and writing directly to an ordinary disk file [23]. This file, called the main database file, is a complete SQL database with a structure that includes tables, indices, triggers and views [23]. The main database file consists of one or more pages; all the pages in a given database structure have the same size [24].

The first page of the main database file contains a 100-byte database header and schema tables [14]. The database header stores file structure information while the schema table contains information about the tables, indices, triggers and views contained in the main database file [14]. The remaining pages are structured as B-trees, where each page contains a B-tree index and B-tree table, which are responsible for storing the actual data [20].

SQLite databases store large amounts of data in multiple tables, but most of the data in these tables are not relevant to event reconstruction. To focus the analysis, the extraction phase identifies and extracts only the relevant tables from SQLite databases. This same technique is applied to the backup database files.

- **Rollback Journal:** A rollback journal [25] is the default method used by SQLite to implement atomic commit and rollback. In a traditional rollback journal, a copy of the original unchanged database content is written to a separate rollback journal file before any changes are made to a database file. In the event of a crash or failure, the content within the rollback journal file is written back to the database file to revert the database to its original state. When the rollback journal file is deleted, a commit occurs that marks the end of the transaction. To improve speed and concurrency, SQLite incorporates a new backup method that uses a write-ahead log (WAL).

- **Write-Ahead Log:** In the case of WAL, no changes are made to a database file; instead, the changes are appended to a separate WAL file. A special record indicating a commit marks the end of the transaction when the record is appended to the WAL file. Moving the pages from the WAL file back to the database is called a "checkpoint." A checkpoint occurs automatically when the WAL file reaches a size of 1,000 pages [25].

  Two files, `<databasefilename> -wal` and `<databasefilename> -shm`, are found in the same directory as the main database file [3].

The WAL file contains the new and updated database pages [3] while the SHM file is the shared memory file associated with the main database [25]. The WAL file contains a 32-byte file header followed by zero or more WAL frames. Since a checkpoint only occurs after 1,000 entries, the WAL file maintains a long record of previously altered and deleted data that can be very valuable in event reconstruction.

3.2 Phase 2: Homogenization

After the SQLite databases have been extracted, differences between the structures and data formats must be reconciled. The second phase, homogenization, reconciles the heterogeneous structures and data formats into a canonical form to allow for efficient analysis [18]. Homogenization involves two steps: structure homogenization and data homogenization. The first step resolves the heterogeneities in the structures of the SQLite databases. The second step transforms the data collected from the databases into a general format.

- **Step 1 (Structure Homogenization):** Structure homogenization facilitates the integration of data from the various SQLite databases. Similar databases from different devices do not always use the same structure. The table structure used by Android devices to store text and multimedia messages differs from the structure used by iPhones. For example, an Android device stores message content in a column named "body" [15] while an iPhone stores the same content in a column named "text" [27]. To support query processing and visualization, which are required during the last two phases, the extracted databases must be transformed to a general structure.

  To achieve structure homogenization, both schema and data type heterogeneity need to be addressed [8]. Schema heterogeneity refers to the structure of the tables that can be different, even if the same type of data is stored. For example, one database may store the first and last names of contacts in separate columns while another database stores the first and last names as a full name in a single column. Although the same type of data is stored, the differences in the schema structure can lead to misinterpretations. To overcome this problem, a single canonical schema must be identified and the data pulled into the canonical schema.

  In addition to the differences found in schemas, different data types are often used to store similar data. For example, a cellphone number can be stored as a string in one database but as a number

in another. To simplify query processing, a single data type must be identified and used to store similar data.

Having achieved structure homogenization, the next step is to reconcile data heterogeneity.

- **Step 2 (Data Homogenization):** As in the case of file structures, differences exist with regard to data formats. For example, timestamps may have different formats, which increases the difficulty of performing analysis and event reconstruction. Thus, timestamps must be converted into a standard format. A standard format such as `YYYY-MM-DD-HH:mm:ss:pppp...` facilitates analysis because it allows the timestamps to be sorted alphabetically and numerically [5].

  Timestamps are also complicated to analyze due to the presence of different time zones and varying clock skews [5]. Thus, the relevant time mechanisms should be examined closely and an appropriate Δ must be determined to limit false positives [5]. Timestamps can be evaluated and ordered using the equation [5]:

$$\forall t_1, t_2, |t_1 - t_2| < \Delta \Rightarrow t_1 \approx t_2. \quad (1)$$

  In addition to differences in timestamps and time offsets, similar logical values are often stored in different formats. This is referred to as value heterogeneity [8]; an example is the representation of true and false as 1 and 0, respectively. To overcome possible confusion and misinterpretation, heterogeneous values must be transformed to canonical forms.

3.3 Phase 3: Schema Integration

After successful data extraction and homogenization, the next step is to integrate the SQLite databases and backup files. Integration, or schema integration, takes multiple schemas as input and produces an integrated schema as output [7].

Schema integration involves four phases [7]: (i) schema translation; (ii) schematic interschema relationship generation; (iii) interschema relationship generation; and (iv) integrated schema generation. Schema translation converts the database schemas into an intermediate canonical representation [18]. Schematic interschema relationship generation identifies objects in the schemas that may be related and categorizes their relationships [7]. Interschema relationship generation describes the interschema relationships [19]. The final phase produces the integrated

schema along with mapping information to support data access via the integrated schema [7, 19].

The output of schema integration is a single, cohesive schema [10], which is also referred to as a "multidatabase" [18]. Schema integration brings together large collections of data, enhancing query processing and simplifying data exploration.

3.4 Phase 4: Exploration

In many cases, potentially useful information may be obtained from an integrated database using simple custom-crafted queries. In a physical environment, location information about an incident such as a gunshot can be acquired from GPS locations recorded on smartphones, cellular tower associations and hotspot connections. Audio and video records could be obtained from smartphones in the proximity of the incident that were in the recording mode. The exact formulation of the query may depend on the density of smartphones in a given location as well as the expected range of the sound. In a tourist hotspot, it is likely that many videos would be recorded.

Clearly, all sources of evidence are not equally precise: a recording may enable an investigator to pinpoint the time very accurately. However, if an individual who was engaged in a telephone conversation heard the sound, the positioning of the gunshot event in time could only be tied to the period of the phone call. Of course, if multiple individuals engaged in phone calls were to hear the gunshot, the intersection of the phone call periods would provide a more accurate estimate of the time. Therefore, an investigator should prioritize queries to initially find recordings that provide the required information, and then proceed to find activities that may yield less precise results.

Queries about average message propagation delays (in mobile networks, Internet and social networks) could be used to infer location information. Querying communications data, such as call logs and messages, can identify mutual relationships between individuals that were previously unknown. Such queries can also determine the propagation of a specific piece of information and identify not only the individuals involved in propagating the information, but also the order in which the propagation took place. Indeed, data exploration using queries can be very useful in myriad scenarios.

Multiple smartphones can help determine the exact times of events more accurately; also they can confirm the correctness of evidence obtained from one (or a few) smartphones if the database entries of all the smartphones are consistent. Conversely, the absence of consistent entries

for a large enough set of independent smartphones where some entries are expected or the presence of inconsistent entries on other smartphones may cast doubt on the evidence.

Despite the utility of simple queries, the real forensic power of a distributed database created from smartphones lies in the complex questions that can be answered. SQL querying and visualization are two important mechanisms that assist in phrasing complex queries and understanding the results.

SQL Querying. SQL can be used to create complex queries that are of forensic use. Considerable flexibility is provided by the WHERE clause of the SQL SELECT statement; indeed, this clause imposes hardly any limits on what may be compared in a distributed database. However, this paper focuses on the powerful abilities of the SELECT statement to summarize results. SELECT can, of course, be used with aggregation functions such as count, sum, average, maximum and minimum. We explore the forensic uses of the GROUP BY, HAVING and ORDER BY clauses.

The GROUP BY clause is often used in conjunction with aggregation functions to group the result set according to one or more columns [13]. From a forensic perspective, the GROUP BY clause provides assistance in categorizing a large collection of data according to specified groups. For example, a table storing text messages from a particular smartphone can quickly grow to a significant size, making it difficult for an investigator to identify the communications that occurred between individuals of interest. The GROUP BY clause can be used to group text messages by individual, reducing the data volume and simplifying the analysis.

The HAVING clause was added to SQL because the WHERE clause could not be used with aggregation functions [13]. Building on the example above, the HAVING clause can be used to select the individuals whose communication rates (average numbers of text messages sent during specific time periods) were particular high. If the period in time correlates with an event being investigated, the individuals with high communication rates could be involved in the event in some manner.

The ORDER BY clause is used to structure data to appear in a specific order [13]. The order in which data is presented can speed analysis. Continuing with the example, the data may be ordered to show the text messages that were sent by individuals near the time when the event of interest was known to have taken place. Identifying these individuals can lead to the discovery of data with forensic value.

Visualization. Visualization techniques can enable an investigator to quickly explore and analyze specific characteristics of the query results [5]; they can also be used to prove or disprove hypotheses. Useful visualization techniques include temporal, relational and dynamic analysis. Part of the power of visualization stems from the fact that a visual representation often reveals patterns, exceptions to patterns and inconsistencies that are not obvious from the raw data. We recommend a two-pronged approach. First, queries may be structured to find artifacts without using visualization. Visualization can then be used to confirm any patterns. Visualization is also used to present complex evidence in court proceedings. In this case, a query could be formulated to test some hypothesis and the result (possibly with visualization) could be used to confirm or reject the hypothesis. The alternative is to "read" the visualizations, i.e., use visualization to formulate hypotheses that are tested in some other manner.

The first visualization technique, temporal analysis, helps reconstruct events using a timeline. In some cases, a partial ordering (or better) suffices; as Cohen [5] points out, it is inconsistent to claim that A caused B if B precedes A. In other cases, exact times may be required.

A query-based strategy to determine a timeline in the given context would proceed as follows. First, a set of smartphones that automatically set their times from external sources (e.g., telecommunications networks or GPS). The consistency of the times on the devices can be confirmed by finding common events in their databases and comparing the recorded times. Care should be taken to ensure that the common events are actually shared events; in many cases, common events may be communicated sequentially to smartphones with expected differences in recorded times. Note also that smartphones that use different shared time servers may have times that are consistent with those that use the same server, but not necessarily with smartphones that use other servers. However, if the time servers are to be trusted, the differences between them should remain constant (i.e., no drift). If this is the case, it is easy to harmonize time.

The next query may then be formulated to determine if the times on the devices containing the evidence can be harmonized with the times on other devices, which is also done by finding common events on the smartphones and the independent events. If this is done successfully, the timeline can be reconstructed using the harmonized time on the devices containing the evidence. Note that the times on different independent smartphones cannot be expected to match perfectly, slight variance is almost always present. This variance can be carried forward during the various queries that set the time of an event on an evidentiary device

not as a specific instant, but as a time range with some probability. In most cases, the ranges would be small enough to identify specific times for evidentiary use; but, of course, these would have to be verified empirically.

Note that the strategy described in the previous paragraph assumes that all the checks will pass. However, this may not be the case in the real world. Hence, the process may be followed by an investigator, with alternatives needing to be explored when the "normal" process fails. More specifically, the investigator may use the process under the supervision of a forensic scientist. When it works (and all standards have been adhered to) the scientist can sign off on the results. This is similar to what happens at a DNA laboratory, where the majority of tests are handled by technicians under the supervision of a forensic pathologist. The forensic pathologist gets involved in the process when the standard protocol does not yield good results.

The second visualization technique, relational analysis, is the identification of meaningful relationships existing between various entities. A query-based strategy to identify the relationships would use custom queries to first select all communications data between a set of smartphones that have evidence pertaining to an event. This data would include text messages, instant messages, email and call logs, all of which could be used to determine relationships between individuals. Next, GPS data or a timeline generated by temporal analysis would be used to identify the relationships between individuals described using times and locations.

Visualizations provided by temporal and relational analyses offer static pictures of events that occurred at a specific times and locations. The third technique, dynamic analysis, can replay a sequence of events by changing one or more properties related to the events. The properties include time and location, which may be changed to visualize the effects on the events. Dynamic analysis provides an interactive view of the reconstruction of the events surrounding an incident. The richer the interactive features, the more thoroughly the events can be explored and analyzed.

It is important to note that the computation of a timeline and the elucidation of relationships, although inspired by visualization, are quite distinct from the visualization process itself. Visualization helps the uninitiated to comprehend the results, but they may read more (or less) in the results than what the results actually convey. Visualization is useful for conveying results that have already been determined to be probative. Visualization is also useful for finding investigative leads. Problems arise when these two applications of visualization are confused.

4. Discussion

The mobile event reconstruction process provides several advantages in support of digital forensic investigations. The process can handle a diverse collection of smartphones, increasing the quality and quantity of evidence. Furthermore, the process allows for the proper integration of datasets extracted from multiple smartphones by leveraging database integration theory. The process also enhances data analysis by using a set of custom queries to analyze the data and reveal latent information. Finally, the process can assist digital forensic investigations by helping visualize the queried data. The output of the process is the reconstruction of events surrounding a particular incident, including the events that occurred before, during and after the incident.

5. Conclusions

Smartphones store vast amounts of valuable data in SQLite databases, making them important assets in digital forensic investigations. A collection of smartphones corresponds to a distributed database of nodes containing potential evidence about events of interest. The mobile event reconstruction process described in this paper allows for event reconstruction by querying an integrated database created from the individual SQLite smartphone databases. The process, which incorporates tried and tested visualization techniques, creates detailed accounts of the events that took place before, during and after an incident.

Future work will focus on refining the mobile event reconstruction process and exploring the addition of evidence from devices other than smartphones. Also, the performance and functionality of the process will be evaluated using real-world scenarios.

References

[1] AccessData, Mobile Phone Examiner Plus (MPE+), Lindon, Utah (`www.accessdata.com/products/digital-forensics/mobile-phone-examiner`).

[2] A. Arnes, P. Haas, G. Vigna and R. Kemmerer, Digital forensic reconstruction and the virtual security testbed ViSe, *Proceedings of the Third International Conference on Detection of Intrusions and Malware and Vulnerability Assessment*, pp. 144–163, 2006.

[3] A. Caithness, The Forensic Implications of SQLite's Write Ahead Log, CCL Group, Stratford-upon-Avon, United Kingdom (`www.cclgroupltd.com/the-forensic-implications-of-sqlites-write-ahead-log`), 2012.

[4] E. Casey, *Digital Evidence and Computer Crime*, Elsevier, Waltham, Massachusetts, 2011.

[5] F. Cohen, *Digital Forensic Evidence Examination*, Fred Cohen & Associates, Livermore, California, 2009.

[6] K. Curran, A. Robinson, S. Peacocke and S. Cassidy, Mobile phone forensic analysis, *International Journal of Digital Crime and Forensics*, vol. 2(3), pp. 15–27, 2010.

[7] A. Elmagarmid, M. Rusinkiewicz and A. Sheth (Eds.), *Management of Heterogeneous and Autonomous Database Systems*, Morgan Kaufmann Publishers, San Francisco, California, 1999.

[8] M. Eltabakh, Data Integration, CS561-Spring 2012, Department of Computer Science, Worcester Polytechnic Institute, Worcester, Massachusetts (`web.cs.wpi.edu/~cs561/s12/Lectures/IntegrationOLAP/DataIntegration.pdf`), 2012.

[9] Forensic Science Central, Crime Scene and Accident Scene Reconstruction (`www.forensicsciencecentral.co.uk/reconstruction.shtml`), 2005.

[10] A. Gal, A. Trombetta, A. Anaby-Tavor and D. Montesi, A model for schema integration in heterogeneous databases, *Proceedings of the Seventh International Database Engineering and Applications Symposium*, pp. 2-11, 2003.

[11] P. Gladyshev, Formalizing Event Reconstruction in Digital Investigations, Ph.D. Dissertation, Department of Computer Science, University College Dublin, Dublin, Ireland, 2004.

[12] P. Gladyshev and A. Patel, Finite state machine approach to digital event reconstruction, *Digital Investigation*, vol. 1(2), pp. 130–149, 2004.

[13] H. Halvorsen, Structured Query Language, Department of Electrical Engineering, Information Technology and Cybernetics, Telemark University College, Porsgrunn, Norway (`home.hit.no/~hansha/documents/database/documents/Structured%20Query%20Language.pdf`), 2012.

[14] S. Jeon, J. Bang, K. Byun and S. Lee, A recovery method of deleted record for SQLite database, *Personal and Ubiquitous Computing*, vol. 16(6), pp. 707–715, 2012.

[15] MD's Technical Sharing, Raw access to SMS/MMS database on Android phones (`minhdanh2002.blogspot.com/2012/02/raw-access-to-sms-database-on-android.html`), February 14, 2012.

[16] J. Morris, *Crime Analysis Charting: An Introduction to Visual Investigative Analysis*, Palmer Enterprises, Loomis, California, 1982.

[17] Oxygen Forensics, Oxygen Forensic Suite, Alexandria, Virginia (`www.oxygen-forensic.com/en`).

[18] M. Ozsu and P. Valduriez, *Principles of Distributed Database Systems*, Springer, New York, 2011.

[19] C. Parent and S. Spaccapietra, Issues and approaches of database integration, *Communications of the ACM*, vol. 41(5), pp. 166–178, 1998.

[20] P. Patodi, Database Recovery Mechanism for Android Devices, Ph.D. Dissertation, Department of Computer Science and Engineering, Indian Institute of Technology Bombay, Bombay, India, 2012.

[21] M. Prasad and Y. Satish, Reconstruction of events in digital forensics, *International Journal of Engineering Trends and Technology*, vol. 4(8), pp. 3460–3467, 2013.

[22] B. Schneier, Attack trees, *Dr. Dobb's Journal of Software Tools*, vol. 24(12), pp. 21–29, 1999.

[23] SQLite, About SQLite, Charlotte, North Carolina (`www.sqlite.org/about.html`).

[24] SQLite, The SQLite Database File Format, Charlotte, North Carolina (`www.sqlite.org/fileformat.html`).

[25] SQLite, Write-Ahead Logging, Charlotte, North Carolina (`www.sqlite.org/wal.html`), 2013.

[26] P. Stephenson, Modeling of post-incident root cause analysis, *International Journal of Digital Evidence*, vol. 2(2), 2003.

[27] The iPhone Wiki, Messages (`theiphonewiki.com/wiki/Messages`), August 16, 2013.

[28] viaForensics, viaExtract, Oak Park, Illinois (`viaforensics.com`).