Temporal Analysis Anomalies with iOS *iMessage* Communication Exchange

Michelle Govan and Kenneth Ovens

School of Engineering & Built Environment,
Glasgow Caledonian University, Scotland.
michelle.govan@gcu.ac.uk
kenneth.ovens@gcu.ac.uk

Abstract. The universal adoption of mobile devices provides an abundance of data for forensic investigators to extract, analyse, and reconstruct events. Unfortunately, anomalies produce misleading temporal data and other discrepancies which, without proper understanding, can hinder investigations. To ensure more data can be converted into reliable evidentiary material this paper presents a detailed study of an Apple iMessage communication exchange in iOS 7, explaining the occurrence of discrepancies and examining temporal data accuracy. The ability to establish a message origin on a system where multiple devices share a single account is also explored.

Keywords: cyber forensics, temporal analysis, iMessage, iOS

1 Introduction

The near ubiquitous use of mobile devices has created individual repositories that provide an abundance of data on a user's activities, which in turn provides investigators with potentially rich sources of information that previously may not have existed. However, such information can only become reliable evidence when there is a complete understanding of the data dynamics on devices that provide such data, and there are explanations for any apparent anomalies that arise.

With reliable temporal data investigators can begin to reconstruct a chronological list of events to find out what happened, when it happened, and who was involved [1]. However, temporal data can also be misleading due to poor configuration, time-zone differences, daylight saving time, clock drift, or how the operating system and application have been programmed [2]. The importance of accurate time sequencing was highlighted by Quick and Choo [3] where anomalies in timings threatened the admissibility of communication evidence in court. Furthermore, erroneous assumptions regarding timings on computers have also led to appeals against verdicts. For example, in Lundy v The Queen [4], specialists were engaged to ascertain if the appellant, as accused, manipulated the system clock on a personal computer to mislead the police investigation.

Understanding the functionality of closed, proprietary software that is installed on digital devices remains a significant challenge, as the code is normally not made available for investigation or research. Consequently, investigators are required to test applications and observe the events that occur in order to understand how the applications function [5].
Previous research by Govan [6] on the temporal exchanges of data associated with Apples iMessage application running on iOS 5, established the origins and dynamics of embedded data, the impact of its varying precision, and explained the occurrence and identification of anomalies. As part of an effort aimed at extending this previous research, this paper presents a detailed study of an iMessage communication exchange on more recent operating systems. The objectives of this study is to: determine the accuracy of time stamps and evaluate anomalies; establish whether the device clock configuration effects evidence reliability; and consequently whether the origin of iMessages could be linked to specific devices.
The paper is structured as follows: Section 2 presents an overview of iMessage database acquisition and structure. Section 3 presents the results of this research and conclusions are drawn in Section 4.

2 Background

This section provides a review of related work, a brief background on Apple's instant messaging service - iMessage, an explanation of the database acquisition, and an overview of the iMessage database structure.

2.1 Related Work

Since their unveiling in 2007, Android and iOS devices have dominated the mobile device market. Subsequently, these devices have been the focus of various research efforts over the past several years. As with any new system, initial research aspired to forensically extract artefacts of interest [7, 8]. This would typically consist of traditional mobile phone data e.g. call logs, contacts, SMS, but with the huge growth in third-party applications such as WhatsApp, Viber, and Facebook Chat, research was required to understand how such equally informative data could be extracted.
Husain and Sridhar [9] made use of an iPhone backup facility to perform a logical acquisition and recovered messages, contacts, and other significant data from AIM and Yahoo! instant messaging applications. Iqbal, et al. [10] researched Samsung's ChatON instant messaging service and not only managed to extract a transcript of a communication exchange, but also documented a schema of the ChatOn database. Levinson et al. [11] used a mock scenario of a missing person investigation to demonstrate that by extracting and analysing data, including geolocation references, recovered from various social media apps, such as Twitter, Facebook, Skype, etc., not only could this information lead investigators to the missing person, but also provide insight into events leading up to such incidents. While the information gleaned from mobile devices can be invaluable

in an investigation, caution must be exercised regarding the accuracy of the associated temporal data.
Other than Govan's [6] research into Apple's instant messaging application iMessage on iOS 5, which revealed anomalies within the temporal data, there is little research into the accuracy of time stamps on modern mobile communications.

2.2 iMessage

iMessage is an instant messaging service from Apple for Mac computers and iOS devices aimed at providing a flexible, confidential, and secure communication system for its users. As well as standard text messages, iMessage supports group messaging, location updates, and attachments such as photos, documents, and videos. Conversations can be started on one device such as a mobile phone, and seamlessly continued on another device such as a tablet or computer. Confidentiality and authenticity are provided by Apple's use of public-key infrastructure (PKI) to encrypt and sign messages and attachments [12].

2.3 SQLite Database

Govan [6] detailed how data relating to iMessage communications can be retrieved from the SQLite database on an iTunes backup of the device, or from `/Volumes/messages/chat.db` on the device itself. The collection of data pertaining to messages sent or received through iMessage are stored in a SQLite database, `sms.db`, which can be retrieved via logical acquisition from the devices media partition `/Library/SMS/sms.db` [7] or the iTunes backup (`sms.db` defined as `3d0d7e5fb2ce288813306e4d4636395e047a3d28` within the backup, derived from the `sha1` value related to the domain, path, and file name of the original file). Utilising structured queries it is possible to extract insightful data from the database [13]. While the structure of the database has changed in iOS 7 the retrieval of data remains the same.

2.4 Date & Time: Structures & Precision

The database, `sms.db`, contains nine tables of which two have relevant temporal data; the ***message*** table and the ***attachment*** table. The ***message*** table contains the message content, temporal data, account detail, globally unique identifiers (GUIDs), and remote party detail. The ***attachment*** table contains temporal data, upload details, and GUIDs as well as file type, size, and location. The configuration of the ***message*** table in iOS 7 differs from iOS 5 and the *'madrid'* internal codename for iMessage has been dropped from the field names. Descriptive temporal data associated with each message can be found in three fields from within the ***message*** table: *date*, *date_read*, *date_delivered*; and from two fields within the ***attachment*** table: *created_date*, and *start_date*.

The date fields are represented in *Mac Absolute Time*, defined as a 32 bit integer of the number of seconds elapsed since midnight on *January 1st, 2001* [14]. The precision of data values within `sms.db` is dependent on whether dates are created locally on the device by the system clock, or received from Apple. Govan [6] established in iOS 5 that if the date is set locally values increment in discrete second intervals, in contrast, if set globally by Apple servers they increment in *128* second intervals. This difference can lead to sequencing discrepancies, but can prove beneficial in identifying potential anomalies.

2.5 iMessage States & Flags

Within the ***message*** table, thirty-five flags identify the state of the message, message direction, and whether the message is plain text or parsed data (URL, email address, etc.). As the message changes state the flag is defined and updated. For example, *handle_id* corresponds to the ***handle*** table, which contains all the user's contact details, and if an iMessage contains an attachment, the *cache_has_attachments* field will be set to '1'. Each field corresponds with a changing status of the message, but these can be defined generally as *undelivered*, *delivered, but unread* and *read*.

3 Results

This section presents anomalies found within iMessage temporal metadata, as well as an evaluation of one-to-one messaging, multiple devices, and iMessage attachments.

3.1 Anomalies

Within digital forensics it is essential to be able to identify the reliability of temporal data that supports or refutes the chronological order of events. However, subtle discrepancies in temporal data created by a shortfall in precision and the mechanisms for inserting data could lead to confusion within forensic analysis and may have a significant impact on the veracity of derived data. For example, a message appearing to arrive on a device before being sent would require an explanation. Unless this anomaly could be explained, it would cast doubt on the reliability of the evidence and subsequently may be ruled inadmissible.
To further explore Govan's [6] discovery that time stamps on iOS 5 devices set globally by Apple servers were only accurate to within *128* seconds, experiments were conducted on iOS 7 and Mac OS X 10.6.8 devices to establish if the anomalies were still present, or identify any new unexplained behaviour. A series of experiments involving 1,800 messages sent from one Apple ID to another were undertaken, each message sent every two seconds. The message database was extracted from both sending and receiving devices and a comparison of time stamps was made.

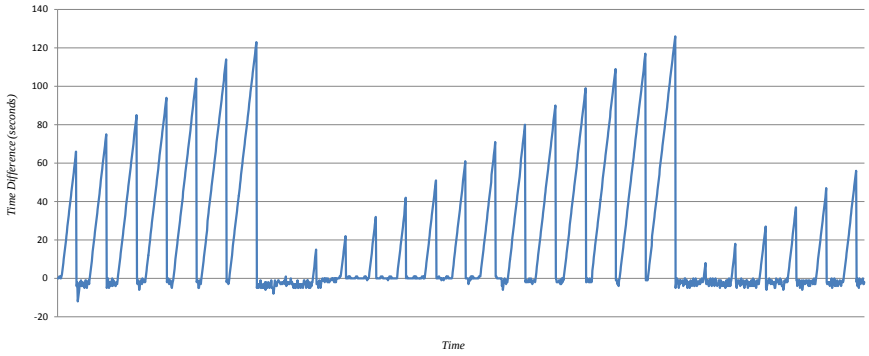

Fig. 1. Time stamp anomalies with device clocks synchronised and accurate.

In contrast to previous studies, time stamps which were considered globally defined e.g. *date_delivered*, exhibit varying reliability in accuracy. In some cases these would be as defined before (e.g. accurate to *128* seconds). However, in other cases, accurately delivered to the nearest second.

By examining the time a message was sent with the time it was defined as being delivered on the receiving device (assuming devices were actively running concurrently), it was possible to establish a recurring pattern of intervals of time accuracy. As depicted in Figure 1, it can be established that there are varying intervals in which the difference is minimal (small differences due to message transfer). While in other intervals the difference is significant, due to the time being defined accurate to *128* seconds, producing clusters of messages that have identical time stamps. Furthermore, the switch between subsequent intervals of accurate time definitions, exhibits a repetition of approximately *137* seconds, which is an unusual phenomenon not generally experienced in wider temporal analysis and difficult to provide a suitable explanation or justification for. However, with time the intervals of accuracy appear to become subsequently smaller, before returning to normal and then subsequently diminishing again.

In order to model this unusual phenomenon it was hypothesised that the relationship between global and local time had a significant impact. Specifically, if global time was in advance of local time the time would be denoted accurately, alternatively approximate to *128* seconds. To verify this hypothesis two further experiments were undertaken. In each case the device clock time was set manually, either in advance or behind time, ensuring that it would always be ahead or conversely behind Apple's server definition of time.

When the receiving device time was defined as being in advance of Apple's server definition, Apple's time definition was exclusively used for time stamping. However, as had been hypothesised, and depicted in Figure 2, while the time stamps increment in *128* second intervals, the time interval at which the time stamps update was approximately *137* seconds. Conversely, when device time

was defined as occurring prior to Apple's time definition, the device time was exclusively utilised.

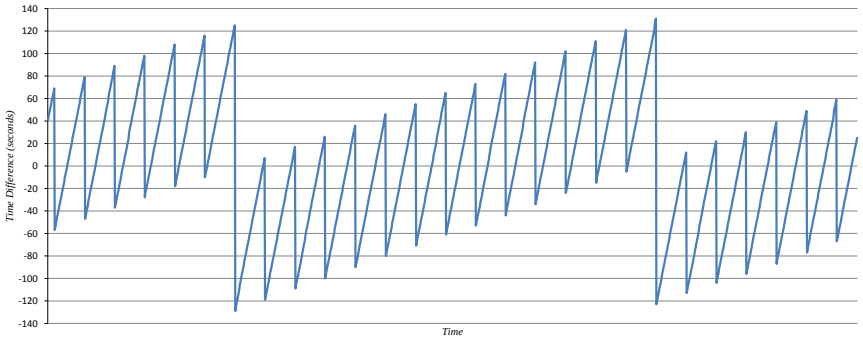

Fig. 2. Receiving device clock set in advance of Apple server defined time.

Consequently, it is clear that the relationship between locally, and globally defined time has an impact on which time is subsequently denoted. If local time is less than global time, then local time accurate to one second is utilised, else it is the global definition which is denoted. Given that global time is defined accurately to *128* seconds, but increments approximately every *137* seconds, this provides the model in which to explain the apparent erratic behaviour in Figure 1.

3.2 One-to-0ne Messaging

This section establishes the origin and dynamics of embedded data within iMessage exchanges. Initially dealing with one to one exchanges and subsequently exploring more complex interactions and the impact of message attachments.

Outgoing Sending iMessages Users initiate new iMessage conversations by entering an email address, phone number, or name. The device contacts Apple servers to obtain encryption keys and routing information to encrypt and deliver the message [12]. These details are included within the ***message*** table along with the following fields: *is_delivered* - defined if the receiving device is online/available; *is_from_me* - defined if the message has come from the user's Apple account; *is_read* - will be defined if the recipient has 'Send Read Receipts' enabled on their account; *is_sent* - is defined if the message is able to reach Apple servers i.e. the device is online.

The temporal data associated with an outgoing message are:

- *date* (denoted by d) - reflects when the message was sent
- *date_delivered* (d_d) - reflects the time a delivered response is received back to the originating device
- *date_read* (d_r) - reflects the time a read response is received (if 'Send Read Receipts' is selected, else undefined)

Table 1. Temporal fields for an outgoing message.

	date d	*date_delivered* d_d	*date_read* d_r
Sender	13:50:00	13:50:02	13:55:26

For outgoing messages, d and d_d are set locally, d_r can be set either locally or by Apple servers. As discussed in Section 3.1, if time stamps are set locally, they will be as accurate as the device clock; if set globally by Apple servers they will be accurate to within *128* seconds. If time stamped locally, d_r would reflect when the device received the read receipt, which typically would be less than one second from when sent from the receiving device. This is reflected in Table 1 which shows the temporal fields associated with outgoing messages populated with typical values.

Incoming Receiving iMessages On the receiving side, the device receives the message from Apple servers and the content is decrypted using the device's private encryption key. iMessages can be queued for up to seven days, for delivery to offline devices [12]. The fields defined for an incoming message include: *is_delivered* - will always be defined as '1' for incoming messages; *is_read* - defined as '1' if read, '0' if not.
The temporal data associated with an incoming message:

- d' - reflects when the message was received, can be set by either the local device or by Apple servers.
- d'_d - undefined, for incoming messages.
- d'_r - reflects when the iMessage application was accessed and is set by the local device.

Table 2. Temporal fields for an incoming message.

	date$'$	*date_delivered* d'_d	*date_read* d'_r
Recipient	13:50:02		13:55:26

The d'_d field is not defined for incoming messages. d' can be defined locally by the device's clock, or by Apple servers, whereas d'_r is always defined by the system clock and will therefore be as accurate as the system clock. This is reflected in Table 2 which shows the temporal fields associated with incoming messages populated with typical values.

Correlation of Temporal Data Govan [6] defined a *'happened-before'* relational model for an iMessage exchange. The primary events were defined as: evt_{send}; $evt_{delivered}$; and evt_{read}; Using the logical expectation that a message is required to be sent before it can be delivered and requires delivering before it can be read, the event relationship was defined as:

$$evt_{send}\ d \longrightarrow evt_{delivered}\ d_d \longrightarrow evt_{read}\ d_r$$

From the 'happened-before' relation model it is possible to create and define rules for the detection of inconsistent events.

3.3 Multiple Devices

iMessage's ability to enable a conversation on one device to be replicated and continued on another, offers users flexibility and convenience. This duplication of data presents digital forensic investigators with opportunities as well as challenges. Its impact within forensics is significant and cannot be ignored.

Multiple Delivery Points As illustrated in Figure 3, a message from one device can be received on multiple devices (endpoints) which share the same Apple ID. The replication and insertion of data can impact on the definition of temporal data values.

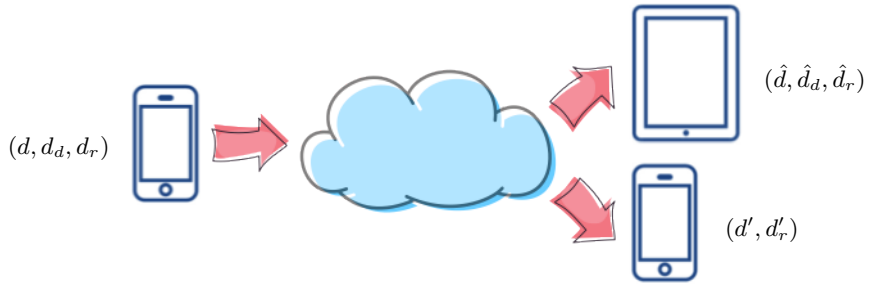

Fig. 3. Multiple endpoints with same Apple ID.

In this scenario the original transition between states and the definition of temporal data for the sending device remain unaltered, the only defining aspect

is that d_d, and where applicable d_r, reflect the first instance a message is delivered or read on a 'recipient' device. Temporal data is not updated if delivered or read activities are performed once more on a secondary 'recipient' device.
Temporal data on the collection of recipient devices will alter depending on the state of each device. When a message is delivered to the device, d' and $\hat{d}$, will reflect the time the message was sent, whereas d'_r and $\hat{d}_r$ provide a reflection of the time the message is viewed within iMessage.

Previous research by Govan [6] based on iOS 5 showed that when a message is read on one device prior to another, the action of viewing is replicated and depicted through equal $\hat{d}_d$ and $\hat{d}_r$ ($\hat{d}_d == \hat{d}_r$), illustrated in Table 3, which defines the time the device received the indication of the message being read (independent of read receipt selection).

Table 3. Temporal values for a message sent to multiple recipients.

	date d	*date_delivered* d_d	*date_read* d_r
Sender	13:50:00	13:50:02	13:55:26
Recipient 1 $'$	13:50:02		13:55:26
Recipient 2 $\hat{}$	13:50:02	13:55:26	13:55:26

Replication of Sent Messages As illustrated in Figure 4, a message sent from one device can be replicated to a secondary device which shares the same account. While the user interface and ***message*** table reflect similar communication exchange, subtle differences in temporal data values and definitions exist, making it difficult to correlate events.

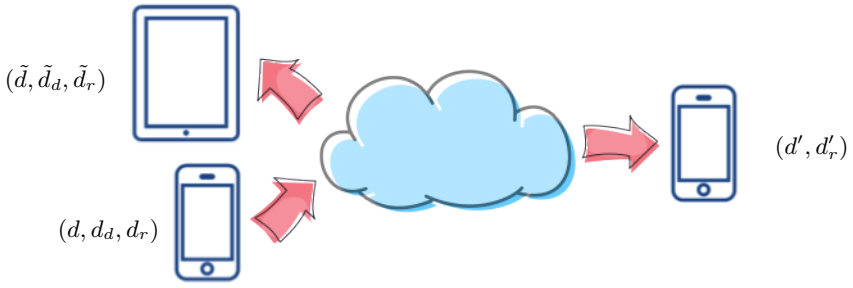

Fig. 4. Replication of a sent message to a secondary device.

In this scenario the original transition between states and the definition of temporal data for the primary sending and recipient devices will remain unaltered, and continue to reflect the model defined in Section 3.2. However, the temporal data on the secondary sending device incorporates deceptive values. As was the case with iOS 5 enabled devices, when a message is replicated to the secondary sending device, $\tilde{d}$ reflects the time the message was sent as would be expected. However, the *is_delivered* field is defined as if the message has been delivered to the recipient even if the message is undelivered. As the original 'Sender' receives a read receipt this temporal data is replicated to the secondary sending device, however the data is also used to populate the $\tilde{d}_d$ field as illustrated in Table 4.

Table 4. Replication of sent messages with read receipts.

	date d	*date_delivered* d_d	*date_read* d_r
Sender	13:50:00	13:50:02	13:55:26
Secondary Sending Device $\tilde{}$	13:50:00	13:55:26	13:55:26
Recipient $'$	13:50:02		13:55:25

In iOS 7, if the recipient does not send read receipts the $\tilde{d}_d$ field is not populated on the secondary sending device as illustrated in Table 5.

Table 5. Replication of sent messages **without** read receipts.

	date d	*date_delivered* d_d	*date_read* d_r
Sender	13:50:00	13:50:02	
Secondary Sending Device $\tilde{}$	13:50:00		
Recipient $'$	13:50:02		13:55:25

In this situation the secondary sending device either holds no data or unreliable temporal data relating to when the message was delivered to the recipient. This imperfection in replication of data could however, assist in determining which device was the original sender of the message. If there are no read receipts sent from the recipient, then only the original sender will have d_d defined.

3.4 iMessage Attachments

The iMessage database contains a separate table composed of metadata relating to message attachments, such as files, images, etc. Attachments are encrypted and uploaded to Apple's cloud computing service, iCloud. Each receiving device on receipt of the message downloads the attachment from iCloud. While the

Table 6. Data from the ***attachment*** **table.**

	created_date	*start_date*	*user_info*
Sender	13:50:00		Encryption & upload data
Secondary Sending Device ˜	13:50:09	13:50:10	
Recipient ′	14:10:02	14:10:03	

creation and sending of messages is replicated throughout each device in the Apple account, uploading of attachments is done once by the original sender.

As illustrated in Table 6, the device that sends the message containing an attachment does not define the *start_date* field. Another distinction is that the *user_info* field is populated with encryption and Uniform Resource Identifier (URI) data relating to the attachment upload. Both the 'Secondary Sending Device' and the 'Recipient' have a defined *start_date* and undefined *user_info*. Consequently, if a message contains an attachment (e.g. image) then by utilising the associated metadata in the attachment table, it is possible to establish whether the device was responsible for the original message.

4 Conclusion

Evidence relating to temporal data is essential when interpreting the sequence and reconstruction of events, but it can be complex and often undocumented. This paper has provided greater understanding of the temporal data associated with iMessage exchanges by: determining the accuracy of time stamps and evaluating anomalies; establishing the impact of clock configuration on reliability; and determining whether the origin of iMessages can be linked to specific devices.

It has been established that the accuracy of iMessage time stamps depends upon whether the time stamping function is carried out globally by Apple servers or the internal system clock of the device. When the internal system clock is used to time stamp, the temporal data is as accurate as the system clock. In contrast, Apple server time stamps appear to be only accurate to *128* seconds, which is further complicated by appearing to only update every approximate *137* seconds, in addition to the uncertainty as to which process executes the time stamp function at any given time.

Seeking to discover the triggers for which process is used to time stamp iMessages, this research has shown that the accuracy of the receiving device's internal system clock influences whether messages are time stamped locally using the internal system clock, or globally using Apple servers. When the internal system clock is ahead of Apple server's definition of time, Apple servers are used exclusively to time stamp iMessages. When the internal system clock is behind Apple server's definition of time, the system clock is used exclusively to time stamp. Consequently, due to the way in which Apple server times are incremented, a device with an accurate system clock will have iMessages time stamped with

both Apple servers and the system clock at varying intervals.
By associating messages with accounts rather than devices, and replicating communication exchange metadata across all devices associated with an Apple ID, the iMessage service does not provide a mechanism within the database metadata to define where activities originated, which in some cases could be essential. By analysing the temporal metadata on all devices in an Apple account, it has been established that in certain situations (e.g. the message contains attachments) it may be possible to define whether a device was the originating device.

References

1. E. Casey, "Chapter 1 - introduction," in *Handbook of Digital Forensics and Investigation*, E. Casey, C. Altheide, C. Daywalt, A. d. Donno, D. Forte, J. O. Holley, A. Johnston, R. v. d. Knijff, A. Kokocinski, P. H. Luehr, T. Maguire, R. D. Pittman, C. W. Rose, J. J. Schwerha, D. Shaver, and J. R. Smith, Eds. San Diego: Academic Press, 2010, pp. 1–17.
2. S. Y. Willassen, "Methods for enhancement of timestamp evidence in digital investigations," Ph.D. dissertation, Norwegian University of Science and Technology, 7491 Trondheim, Norway, Jan 2008.
3. D. Quick and K.-K. R. Choo, "Forensic collection of cloud storage data: Does the act of collection result in changes to the data or its metadata?" *Digital Investigation*, vol. 10, no. 3, pp. 266–277, Oct. 2013.
4. "Lundy v the queen (new zealand) [2013] UKPC 28," Oct. 2013. [Online]. Available: http://www.bailii.org/uk/cases/UKPC/2013/28.html
5. B. D. Carrier and E. H. Spafford, "Defining event reconstruction of digital crime scenes," *Journal of Forensic Sciences*, vol. 49, no. 6, p. 1291 to 1298, Nov. 2004.
6. M. Govan, "Temporal analysis of an iMessage communication exchange," in *Cyberforensics Perspectives*, Cardiff, Wales, Jun. 2013, p. 81 to 88.
7. J. Zdziarski, *iPhone Forensics: Recovering Evidence, Personal Data, and Corporate Assets*, 1st ed. Sebastopol, CA: O'Reilly Media, Sep. 2008.
8. J. Lessard and G. Kessler, "Android forensics: Simplifying cell phone examinations." *ECU Publications Pre. 2011*, Jan. 2010. [Online]. Available: http://ro.ecu.edu.au/ecuworks/6479
9. M. I. Husain and R. Sridhar, "iForensics: forensic analysis of instant messaging on smart phones," in *Digital Forensics and Cyber Crime*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, S. Goel, Ed. Springer Berlin Heidelberg, Jan. 2010, no. 31, pp. 9–18.
10. A. Iqbal, A. Marrington, and I. Baggili, "Forensic artifacts of the ChatON instant messaging application," in *8th International Workshop on Systematic Approaches to Digital Forensic Engineering*, Hong Kong, Nov. 2013.
11. A. Levinson, B. Stackpole, and D. Johnson, "Third party application forensics on apple mobile devices," in *2011 44th Hawaii International Conference on System Sciences (HICSS)*, Jan. 2011, pp. 1–9.
12. "ios security," Feb 2014, White Paper. [Online]. Available: http://images.apple.com/ipad/business/docs/iOS_Security_Feb14.pdf
13. J. Zdziarski, *Hacking and Securing iOS Applications: Stealing Data, Hijacking Software, and How to Prevent It*, 1st ed. O'Reilly Media, Jan. 2012.

14. “Mac developer library: Time utilities reference,” https://developer.apple.com/library/mac/documentation/corefoundation/reference/cftimeutils/Reference/reference.html, accessed: 2013-03-13.