Contents lists available at ScienceDirect

Digital Investigation

journal homepage: www.elsevier.com/locate/diin

Android forensics: Interpretation of timestamps

M. Kaart[*], S. Laraghy

Netherlands Forensic Institute, Dept. Digital Technology and Biometrics, Digital Technology Group, PO Box 24044, 2490 AA The Hague, The Netherlands

ARTICLE INFO

Article history:
Received 20 February 2014
Received in revised form 5 March 2014
Accepted 1 May 2014
Available online 2 June 2014

Keywords:
Android
Forensics
settings.db
mmssms.db
Clock skew
Timestamp
Time zone

ABSTRACT

Interpretation of traces found on Android devices is an important aspect of mobile forensics. This is especially true for timestamps encountered on the device under investigation. In the presence of both naive and UTC timestamps, some form of timestamp normalisation is required. In addition, the investigator needs to gain some understanding of potential clock skew that may exist, especially when evidence from the device under investigation has to be correlated to real world events or evidence from other devices. A case study is presented where the time zone on the Android device was set incorrectly, while the clock was set to correspond to the time zone where the device was actually located. Initially, the fact that both time zones enforced daylight saving time (DST) at different periods was expected to complicate the timestamps normalisation. However, it was found that the version of the Time Zone Database on the device was outdated and did not correspond to the actual time zone rules for the given period. After the case study, the results of experiments on a broader range of devices are presented. Among other things, these results demonstrate a method to detect clock skew based on the `mmssms.db` database. However, it was also found that the applicability of this method is highly dependent on specific implementation choices made by different vendors.

Introduction

Many of the Mobile Forensics cases handled at the Netherlands Forensic Institute are related to the acquisition of flash data or to decoding a specific file type or other artefact due to lack of support in common (commercial) tools. However, a somewhat different category of cases are those in which the acquisition or decoding of data is not the primary concern. These cases are often referred to by the rather broad term data-analysis.

In such cases, a public prosecutor or judge is interested in questions such as "When was this device last connected to a wireless network?" or "Was this (child pornographic) image in the WhatsApp media folder the result of sending or receiving a message?". Other examples are "When was this device last shut down or disconnected from the mobile network?" and "Where was the device located at the specific time of the crime?".

While finding specific data or decoding specific file types is an important part of any investigation, the mere presence of a picture or a GPS coordinate, for example, is often not enough to answer such questions adequately. Instead, it is necessary to combine multiple traces and to reason about the circumstances that may lead to the specific combination of traces on the device under investigation.

More often than not, to be able to reason about such scenarios it is important to have a good understanding of some fundamental aspects of the device and its operating system and of the consequences these aspects have on interpretation of the evidence. One such aspect, and the main topic of this work, is that of time and date. It is

[*] Corresponding author.
E-mail address: m.kaart@nfi.minvenj.nl (M. Kaart).

http://dx.doi.org/10.1016/j.diin.2014.05.001
1742-2876/© 2014 Elsevier Ltd. All rights reserved.

important to understand time and date related settings and the consequences these settings have on the interpretation of timestamps encountered on Android devices.

The goal of this article is two-fold. First, it aims to increase awareness in the mobile forensics community that proper understanding of timestamps should be part of any investigation. This is illustrated by a case example in which a specific combination of settings led to the situation where timestamps on an Android device could be easily misinterpreted. The second goal is of a more practical nature: it provides results of several experiments relevant to interpretation of timestamps on Android devices. The experiments were performed on a range of Android devices produced by different vendors, running different versions of the operating system.

Relevance

The relevance of Android Forensics is hardly an issue for discussion. Hence, this section will not include any, probably quickly outdated, statistic on current market share of Android devices. Instead, the importance of proper interpretation of timestamps in digital forensics is briefly discussed here. While the case study and experiments presented here involve Android devices, time and date in any area of digital forensics is often a complex matter, as already recognised by Boyd and Forster in 2004. The relevance of any work related to interpretation is emphasised by the fact that most literature on Android Forensics seems to focus on acquisition, malware, and on some special classes of applications, as discussed in section Related work. No references were found that had the interpretation of timestamps on Android devices as its primary focus.

Forensic investigations often involve a time and date component. In some cases, the traces on a device need to be correlated to the time at which some real-world event took place, and in other cases traces from multiple systems need to be correlated. But even when absolute time is not an issue, timestamps provide the investigator with a means to infer chronological ordering of traces.

The availability of a timeline option in many of the (commercial) tools makes this part of the digital forensic process seem dangerously straightforward. Dangerously, because it is often the basis for the rest of the investigation and once the timeline analysis is completed, the timeline is often taken as ground truth upon which further analysis will be based. The risks of this approach are clearly demonstrated by a case example described by Boyd and Forster (2004). They state, and rightfully so, that the availability of tools cannot be a substitute for proper understanding of the evidence by the investigator.

When timeline analysis and interpretation of timestamps is completely performed by some 'behind the scenes' tool, this may result in the situation where naive timestamps[1] from the FAT file systems sit happily next to UTC-based timestamps from other file systems and the investigator is unaware of the time adjustment that should have been performed. Or rather, unaware *whether* any time zone corrections should have been performed, as the case study presented in section Case study will illustrate.

Related work

Barmpatsalou et al. (2013) performed a review of 7 years of mobile device forensics. This review contains many references to publications in the area of Android device forensics, which will not be repeated here. Of course, Hoog (2011) provides the necessary background knowledge that is needed for any forensic investigator starting in the area of Android Forensics.

Most of the publications on Android Forensics focus on acquisition of flash (Vidas et al., 2011; Simao et al., 2011) or RAM (Thing et al., 2010; Sylve et al., 2012), on the exploration of some previously unexplored file system (Fairbanks, 2012; Quick and Alzaabi, 2011) or on a specific class of applications such as cloud storage (Chung et al., 2012) or social networking applications (Mutawa et al., 2012). Another area that gains much attention is that of analysis and detection of malware (Guido et al., 2013).

Progress in all these areas is highly important, but for a case such as the one described in section Case study, the added value of such work is limited. There is often a need to understand fundamental properties of the Android operating system itself, which is provided by works from Hoog (2011) and Lessard and Kessler (2010) and in case examples such as described by Racioppo and Murthy (2012).

However, also in works covering more fundamental areas of Android forensics, the descriptions are often limited to the indication of interesting data locations. To illustrate: Hoog (2011), Vidas et al. (2011), Lessard and Kessler (2010), Racioppo and Murthy (2012) and Simao et al. (2011) all direct the reader to the `mmssms.db` database in the path `/data/data/com.android.providers.telephony/databases`, but only the first author gives the reader more detail than just the file's location. Hoog (2011) mentions some of the tables and states that dates and times are stored in this database as 'Unix Epoch in milliseconds'. Other important information, such as whether or not raw information from SMS PDUs[2] is stored in the database, how records should be interpreted and whether timestamps are derived from the internal clock or from some external time source such as the SCTS field[3] in the PDU, was not found in the literature.

A similar observation applies to literature on common Android settings, such as those stored in the database `settings.db` located in the path `/data/data/com.android.providers.settings/databases/`. Simao et al. (2011) mention this data location, but without any detail on how to interpret the records in this database.

Only one reference was found that mentioned the location of the time zone setting. Quick and Alzaabi (2011) state that the file `/data/property/persyst.`

[1] A naive timestamp is defined here as a timestamp for which the time zone in which it is stored is either not recorded, or is unknown.

[2] PDU (Protocol Description Unit) is a binary format used for encoding SMS messages.

[3] SCTS (Service-Centre-Time-Stamp) is a field in the PDU that contains the time of arrival of the message at the SMS Service Centre (ETSI, 1998b).

`sys.timezone` contains the time zone for the device under investigation.

As mentioned earlier, Boyd and Forster (2004) present an insightful case example that illustrates the importance of proper time zone adjustments. In their work, which is oriented towards Windows forensics, they provide the readers with a checklist that can be used when dealing with date and time issues. This includes determining the time zone and any potential daylight saving time adjustments, checking whether time structures stores time in local time or UTC, seeking external timestamps (dubbed corroboration by the authors) to increase understanding of proper interpretation of timestamps and testing results on reference devices.

Buchholz and Tjaden (2007) explain the importance of determining clock skew when correlating traces from multiple systems or relating events to the 'real' time. They measured that only 74% of the observed hosts were within a 10 second margin of the reference time, stressing the need for more research in the area of 'forensic time' and 'clock analysis'.

Organisation of this article

The remainder of this article is structured as follows. It begins in section Time and date issues with a discussion of time and date issues. Then, in section Case study a case example is given to illustrate the importance of time and date related settings for a proper interpretation of the timestamps in the evidence. This is followed in section Experiments by a description of additional reference experiments that were performed. The results of those experiments are discussed in sections Time and date related settings & automated time synchronization, Timestamps in mmssms.db and Time zone implementations. The work is discussed in section Discussion and the conclusion and ideas for future work are presented in section Conclusions and future work.

Time and date issues

The issues pertaining to time and date can roughly be divided into two main areas. The first main issue is detection of, and possibly compensation for, clock skew. Clock skew is defined here as the amount of time units (often seconds or milliseconds) by which a clock deviates from the 'real' time. The number of problems this introduces into a forensic investigation depends on the level of precision that is required. Buchholz and Tjaden (2007) explain this in great detail and they state that even for NTP-synced systems, problems remain when sub-millisecond resolution is required. The other main issue is that of normalisation of timestamps in such a way that they all represent the time in the same time zone. Both issues will be briefly explored in relation to Android forensics here.

Detection of clock skew

For reasons described earlier, one of the first steps in any investigation should be to assert if the device's system clock was registering the correct time. When the device is still operational, the clock skew may be observed by comparing the displayed time with the time of an NTP-synced computer. However, the actual device is not always available or operational at the time of a digital investigation. Also, the fact that the system clock was running correctly at acquisition time is no guarantee for the accuracy of the system clock at the time period relevant to the investigation.

Automatic time synchronisation

Mobile devices often have a mechanism to keep their clocks synchronised to some network provided time source. In general, three methods of clock synchronisation are perceived. The first option would be that a device uses the Network Time Protocol (NTP, Mills et al., 2010) to correct its internal clock. For this, a data connection either through Wi-Fi or the mobile network is necessary. Another possibility is that time information from the mobile network through a mechanism called Network Identity and Time Zone (NITZ) (ETSI, 1998a) is used. However, this is an optional part of the GSM standard and not all mobile carriers implement this on their networks. A third perceived option may be that a device sets its internal clock based on the signals received from the GPS system. In some cases it may be necessary to know which methods are actually used and an initial experiment related to this is described in section Time and date related settings & automated time synchronization.

The automatic time synchronisation setting can be enabled or disabled on Android devices. In a forensic investigation, it is a good starting point to check whether this option is enabled or disabled and when the setting was last changed. The case study in section Case study gives an example of this and contains pointers to relevant data locations for this setting. However, as experiments in section Time and date related settings & automated time synchronization demonstrate, the fact that the automatic time synchronisation is enabled is no guarantee that the clock is actually synchronised. This also depends on other settings and circumstances.

External timestamps

A common method to detect clock skew is to identify sources of timestamps that are generated outside the device under investigation. By comparing these timestamps to timestamps which are known to be generated based on the devices' internal clock, potential clock skew can be identified. Note that the amount of clock skew may vary over time, due to clock imperfections or clock-manipulation as an anti-forensics measure. Therefore it is often desirable to obtain an indication for clock skew during the period of interest for the criminal investigation. The success of such efforts depends on many factors, such as the required precision and the availability of traces with external timestamps.

One such external source of time are call detail records. In some investigations, the police have acquired call detail records from the mobile network carrier. Timestamps in these reports can be correlated to call-logs and SMS messages from within the device itself. For Android devices, call logs may be located in databases such as `contacts2.db` in the path `/data/data/com.android.provider.contacts`.

External records such as call-records or server-logs are not always available, and in such cases one has to rely on the availability of externally generated timestamps that are present from within the device itself. An example of this is given in the case example in section Case study, where SMS messages notifying the user of a missed call or of a voicemail message contained a timestamp within the message text. These timestamps could be compared with timestamps present in the SMS database `mmssms.db`, mentioned earlier. Other methods include the use of timestamps in downloaded files, timestamps present in website (fragments) or the use of the expiry time for cookies, although the reliability of this latter method remains to be determined (Minnaard, 2013).

Normalisation of timestamps

Normalisation of timestamps requires a proper understanding of the various formats and time zones in which timestamps are stored. For some traces this is well-known, such as the timestamps in the various file systems' structures. For example, Ext2 and 3 store their timestamps as Unix Time, which is defined as the total number of seconds since 01-01-1970 00:00:00 UTC (Carrier, 2005). Ext4, which is becoming prevalent in Android devices, adds nano-second precision to this for some of the timestamps (Fairbanks, 2012). External media cards in Android devices are often formatted with the FAT32 file system on which timestamps are stored in local time (Carrier, 2005).

Note that recent versions of the vfat driver support an additional mount option `tz = UTC`. The documentation in the Linux kernel source tree (Linux Kernel Organization, Inc, 2013) states that this option causes timestamps to be interpreted as UTC rather than local time. The exact consequences of this mount option for timestamps generated on the Android device and stored in FAT file systems remains future work, but it is advisable to check whether this mount option is enabled when investigating Android devices.

In order to correlate timestamps of both types (local and UTC), timestamps have to be translated to a common time zone, based on the time zone settings on the device under investigation. As mentioned earlier and demonstrated in the case study in section Case study, the current time zone setting can be found in the file `/data/property/persist.sys.timezone`.

Normally, knowing the time zone in use would be enough to perform the necessary timestamp conversions. However, as will be illustrated by the example in section Case study, time zone implementations may be flawed and this has consequences for the normalisation of timestamps. In addition, for timestamps within databases or other files it may be necessary to perform some basic experiments to determine the time zone in which the timestamps are stored. It should also be noted that time zone settings may change over time, so it is advisable to check the time when this setting was last altered.

Case study

In this section, a case study is presented that illustrates the difficulties that may arise when timestamps are to be normalised for further analysis. It illustrates the importance of knowing under which conditions the timestamps are generated (clock setting, time zone setting, time zone implementation). This case exposes a slightly different problem with time zone conversions than the problem that was described in Boyd and Forster (2004). Note that some details were left out and some factual data, such as exact timestamps and message contents, were modified as a means to anonymise the case for this article.

Introduction

The case involved an Android device similar to reference device 1, which was used as reference model during the investigation. See section Reference devices for a description of the reference devices used in this work.

The device was found during a police investigation and it was known from additional information that the device had become inactive, or at least disconnected from the mobile network, during a certain period of time. One of the goals of the forensic investigation on this device was to determine the exact moment at which the device had become inactive or disconnected. It was important to obtain a precision of minutes in order to reconstruct certain events in the investigation.

DropboxManagerService

One of the challenges was to find traces of the deactivation or disconnection from the mobile network. It emerged that there were several traces indicating that the device had been rebooted and had become stuck in the pin-entry dialogue afterwards. The exact combination of traces that led to this conclusion are outside the scope of this article, but one type of trace is worth mentioning.

On Android devices, the folder `/data/system/dropbox` is used by a service known as DropBoxManagerService.[4] This mechanism is used to persistently store 'chunks of data (from various sources – application crashes, kernel log records, etc.)' (Android Open Source Project, 2014c). Please note that this is totally unrelated to the DropBox cloud storage service.

It was found during reference experiments that one of the files created in the location `/data/system/dropbox` has a name with the following file name pattern: `SystemBoot@[timestamp].[lost|txt]`. The experiments showed that this type of file is consistently generated at boot time and that the timestamp part of the file name reflects the time at which the device was booted (Unix time with millisecond precision in UTC). Fig. 1 shows an example of its contents from reference device 1 (the last line is shown in two lines for space considerations).

Another interesting file type that may exist in this location are files that have the following file name pattern: `event_data@[timestamp].txt`. Although their exact

[4] This path was verified by inspecting the source code file `SystemServer.java` in the source code branch of the 4.0.3 release. See Android Open Source Project (2014b) for instructions on downloading the Android source code.

```
Build: SEMC/X10i_1234-8547/X10i:2.3.3/3.0.1.G.0.75/tB_P:user/release-keys
Hardware: es209ra
Bootloader: unknown
Radio: unknown
Kernel: Linux version 2.6.29-00054-g5f01537 (SEMCUser@SEMCHost)
        (gcc version 4.4.3 (GCC) ) #1 PREEMPT Sun Jun 19 22:24:46 2011
```

Fig. 1. Contents of the file `SystemBoot@1390913132201.txt` acquired from reference device 1.

meaning has yet to be determined, reference experiments showed that these files were generated consistently at 30 min intervals. This fact could be used to obtain a good indication of consecutive periods in which the device was operational.

The fact that the Yaffs2 file system (Manning, 2012) was in use provided the investigation with many deleted versions of these files that might not have been otherwise available. Reference experiments on the devices enumerated in section Experiments show that both types of traces described here, occur on all tested versions of the Android operating system, although persistence of such traces over longer periods of time has not yet been examined.

Time and date related settings

The other challenge was that of finding the exact time at which these and other traces had been generated. Since the investigation involved correlation with real-world events, determination of clock skew was of high importance. But also, proper interpretation of timestamps generated on the device turned out to be a major component in the investigation. Reference experiments showed that several data locations contained relevant information concerning time and date settings.

persist.sys.timezone

This file, located in the path `/data/property/` contains the time zone that is currently in use. Reference experiments showed that the modified time on this file indicates that the time zone has not been altered since that time. This file on the case device contained the following time zone: `Africa/Casablanca`. All timestamps on the file were in 2012, whereas the relevant period of the investigation was in 2013.

com.android.settings_preferences.xml

This file, located in the path `/data/data/com.android.settings/shared_prefs/`, contains several fields that are related to date and time settings. Most of these are related to time representation, but the field `auto_time` was found to be related to automatically synchronising the system clock to some network provided time. This file on the case device contained the following: `<boolean name="auto_time" value="false"/>` and reference experiments showed that this indicates that automatic time synchronisation was disabled. The timestamps on this file were from after the relevant period, but based on metadata and file contents of previous versions within that were still available in the Yaffs2 file system, it was inferred that this specific setting had not been changed since 2012.

settings.db

This file, located in the path `/data/data/com.android.settings/databases/` contains a wealth of information pertaining to the settings and operational properties of the device. This includes frequently changing settings, such as the screen brightness or the ring volume. Reference experiments showed that changing any of these parameters leads to an update of the file. This indicates that the last modification timestamp of the database file itself is not a proper indicator for the time at which a specific settings was last altered.

For this case, the main interest lies in settings related to network access and date and time settings. See section Time and date related settings & automated time synchronization for an experiment related to these settings on multiple devices. The main record of interest related to time and date settings is located in the `system` table in a record with the name `auto_time`. Experiments indicated that this setting was related to automatically synchronising the system clock to some network provided time, similar to the field with the same name in `com.android.settings_preferences.xml`. In fact, it was found that both these values toggled at the same time when altering this setting through the Android settings menu.

The value of this record in the `settings.db` database on the case device was determined by exporting the database with the SQLite `.dump` command. The corresponding line in the SQLite export was: `INSERT INTO "system" VALUES(2383,'auto_time','0');`. The value 0 indicates that automatic time synchronisation is disabled on this device.

The first field in this table is the so called `_id` field which is the table's primary key. This field uses an autoincrement integer value, a fact that can be exploited in forensic investigations of such databases, as will be discussed in section Time and date related settings & automated time synchronization. Here too, multiple previous versions of the file were available and by comparing dumps of older versions it was found that this setting had not been altered since 2012. This was supported by the fact that the auto-increment `_id` field for this setting was constant across all recoverable versions of the `settings.db` database.

Intermediate conclusions

From the date and time related setting the following was concluded:

- The time zone was set to Africa/Casablanca
- Automatic time synchronisation was disabled

```
You have a notification for a missed call
  from +316xxxxxxxx on 2013-11-09 10:22:31
```

Fig. 2. Anonymised and translated version of SMS message encountered on device under investigation.

- These settings were last modified prior to the relevant period of the investigation.

At this time it should be noted that the device was located in the Netherlands and that it was not yet clear whether the system time was actually set in relation to the Dutch local time (in which case the time zone setting would be wrong) or to actual Africa/Casablanca time (in which case the clock would be one or two hours off from the Dutch local time, depending on the time of year).

Clock skew detection based on mmssms.db

During the investigation, it was noted that several SMS messages existed that contained a timestamp within the message text. It emerged that these messages were notification from the mobile operator that there were missed calls and/or voicemail messages. An example message (translated to English, timestamps altered and number anonymised) is given in Fig. 2.

It was determined by reference experiments with a SIM card from the same mobile carrier, that such messages were indeed generated when a call was not answered. The exact conditions under which such messages were generated were not exhaustively studied, but these may include signal loss or deliberately not answering a message. However, it was established that the timestamps within the message text were based on the Dutch local time and that they were unrelated to the actual settings of the devices system clock. This fact provided the investigation with an externally generated timestamp that could then be compared with internally generated timestamps.

```
CREATE TABLE sms (
  _id INTEGER PRIMARY KEY,
  thread_id INTEGER,
  address TEXT,
  person INTEGER,
  date INTEGER,
  protocol INTEGER,
  read INTEGER DEFAULT 0,
  status INTEGER DEFAULT -1,
  type INTEGER,
  reply_path_present INTEGER,
  subject TEXT,
  body TEXT,
  service_center TEXT,
  locked INTEGER DEFAULT 0,
  error_code INTEGER DEFAULT 0,
  seen INTEGER DEFAULT 0,
  semc_message_priority INTEGER,
  parent_id INTEGER,
  delivery_status INTEGER);
```

Fig. 3. CREATE TABLE statement for the sms table in mmssms.db the device under investigation.

Table 1
DST changes for CET and WET time zones in 2013 (Time and Date AS, 2014).

Time (UTC)	Europe/Amsterdam	Africa/Casablanca
2013-01-01T00:00:00Z	UTC+1	UTC
2013-03-31T01:00:00Z	UTC+2 (DST start)	UTC
2013-04-28T02:00:00Z	UTC+2	UTC+1 (DST start)
2013-07-07T02:00:00Z	UTC+2	UTC (DST end)
2013-08-10T02:00:00Z	UTC+2	UTC+1 (DST start)
2013-10-27T01:00:00Z	UTC+1 (DST end)	UTC+1
2013-10-27T02:00:00Z	UTC+1	UTC (DST end)

The database `mmssms.db`, located in `/data/data/com.android.providers.telephony/databases`, was examined in order to determine whether the records contained internally or externally generated timestamps. Fig. 3 shows the `CREATE TABLE` statement that is found for the sms table when exporting the `mmssms.db` from this device with the SQLite `.dump` command. It was determined with reference experiments that the date-field contains a unix time timestamp in UTC, which reflects the moment at which the message is received. In addition, by manipulating the device's system clock prior to the reception of a SMS message, it was determined that the timestamp was generated based on the internal system clock and not on some external value such as a timestamp from the SCTS timestamp in the PDU.

The timestamp in the sms table for the message in Fig. 2 was recorded as `138399256958`. Translating this to a human readable timestamp in the ISO-8601 format yields the following: 2013-11-09T10:22:49Z, which corresponds to the timestamp recorded within the SMS body text, give or take a few seconds. The same applied to all other messages of this kind present in the `mmssms.db` database. Since it is known that the timestamp within the message body text is recorded in Dutch local time and that the Africa/Casablanca time zone has a zero hour offset to UTC in November 2013 (see Table 1), it was concluded that the system clock was set to correspond to the Dutch local time.[5] This may be the reason that automatic time synchronisation was disabled on this device.

Without further analysis, it was impossible to infer whether or not the clock was accurately representing the Dutch local time during its entire existence, but it was determined that as similar SMS messages were present for the entire relevant period of the investigation, that all these messages confirmed the same combination of time zone and clock setting.

[5] This was confirmed by other traces not mentioned here. Examples of these are the files that are placed onto the Android device during acquisition by the UFED acquisition device. In this process, files were placed in the path `/data/local/tmp`, such as `nandread_1.3.4382` and `fourrunnerStatic_1.3.4382`. It emerged that these files had two of their timestamps derived from the UFED Touch's internal clock, which provided us with additional external timestamps.

Normalisation of timestamps

The device had an SD card which contained a FAT32 file system and internal NAND flash which contained several Yaffs2 file systems. It is known from the literature that FAT32 timestamps are based on the current system clock, without any time zone conversions. The Yaffs2 file system timestamps are stored as Unix time (Quick and Alzaabi, 2011), which is expressed in seconds since Unix epoch as mentioned earlier. This behaviour was confirmed by several experiments, including manipulation of the system clock and making a photograph with the devices internal camera of a clock that showed the accurate (NTP-synced) Dutch local time.

In order to normalise timestamps from both file systems, information about the Africa/Casablanca time zone was sought. In 2013, according to public sources, the Africa/Casablanca time zone (also identified as Western European Time or WET in short) employed daylight savings time (DST) in 2013 (Time and Date AS, 2014). An interesting detail is that in 2013 DST was suspended during the Ramadan religious event, making time zone normalisation potentially more complex.

Table 1 shows the DST changes that were active for the Africa/Casablanca time zone in 2013. The table also includes the DST changes for the Europe/Amsterdam (CET) time zone. Each row contains the start or end time of DST in one of the time zones, with the exception of the first row, which is merely included to indicate the UTC-offset at the start of the year 2013.

With the DST periods known from Table 1 and under the assumption that the user of the device kept the system time manually in sync with the Dutch local time, the values of both UTC and naive timestamps that would be stored at a specific moment in time can be inferred. Table 2 shows the time-part of the timestamps that would be generated at various moments in 2013. The first column contains the actual UTC-time, whereas the second column contains the clock-setting manually adjusted by the user to match Dutch local time. The third and fourth columns contain the naive and UTC timestamps that would be generated when the Africa/Casablanca time zone is in use.

Normally, this knowledge would have been enough to normalise all timestamps and to represent these in the same time zone, mostly UTC. However, another interesting fact was observed. The relevant period of this investigation was during a period at which DST was not active in the Africa/Casablanca time zone. Hence, the UTC offset was +0 h. As can be seen in the last row of Table 2, in this case, both naive and UTC timestamps actually represent Dutch local time.

Table 2
Timestamps stored when time zone would be accurately implemented.

Actual time (UTC)	System clock	Naive timestamps	UTC timestamp
2013-01-01T12:00:00Z	13:00	13:00	13:00
2013-04-01T12:00:00Z	14:00	14:00	14:00
2013-05-01T12:00:00Z	14:00	14:00	13:00
2013-08-01T12:00:00Z	14:00	14:00	14:00
2013-09-01T12:00:00Z	14:00	14:00	13:00
2013-11-01T12:00:00Z	13:00	13:00	13:00

However, for completeness some additional tests were performed. The date on the reference device was consecutively set to the first of each month in 2013 and any changes in the system clock were observed. For the Dutch time zone Europe/Amsterdam we had observed that the system clock would automatically adjust to account for DST, but when the time zone was set to Africa/Casablanca, no such compensation occurred. Apparently, the issue of proper normalisation was not completed yet.

Time zone database

The observations described above led to the insight that the time zone implementation on the device did not properly reflect the current status of the time zone rules in the Africa/Casablanca area. As a consequence, normalisation of the timestamps based on the currently active and published rules of the time zone, might result in erroneous normalisation of the timestamps.

This, in turn, lead to a further examination of the time zone database that was used on both the reference device 1 and on the case device. The path `/system/usr/share/zoneinfo` contains three files that all start with the string `zoneinfo`. This location stores the Time Zone Database (IANA, 2014) that is in use on the device. The file `zoneinfo.version` indicates the version of the Time Zone Database and both case device and reference device 1 contained the value `2010k` here.

The Time Zone Database files were stored in some binary format that was not studied in more detail. Instead, the source for this version of the time zone database was fetched from IANA's public ftp server[6] and the contents of the file `africa` was studied. It was found that the rules corresponded exactly with the observed behaviour of the reference device. But, as will be discussed in section Time zone implementations, the rules were no longer representative for the actual DST rules enforced in the Africa/Casablanca time zone.

Conclusion

Due to the fact that the time zone database on the device under investigation was inaccurate for the actual DST changes in the year 2013, the approach where the publicly available time zone information would have been applied in order to normalise the timestamps would have led to incorrect conclusions. As it turned out, the fact that the active Time Zone Database on the case device did not enforce DST changes in 2013 at all, simplified timestamp normalisation considerably. It lead to the situation where both the naive timestamps and the UTC timestamps both represented the local time on the device for every timestamp during the period for which the Africa/Casablanca time zone was active. Based on the `mmssms.db` analysis it was shown that this corresponded to Dutch local time.

[6] See: ftp://ftp.iana.org/tz/ (Last Accessed February 2014).

Experiments

As was already mentioned by Barmpatsalou et al. (2013), it is generally accepted that even devices that run the same version of an operating system can present different behaviour. This prompted further research in order to check whether the observations presented in the case study in section Case study also hold for other combinations of device brands and other versions of the Android operating system. This was also triggered by the fact that in another case it was found that the timestamp in the `sms` table of the `mmssms.db` database was not set based on the internal clock, in contrast to what was found on the device in the case study.

Reference devices

The approach taken was to perform a series of reference experiments on a set of devices covering the most frequently used Android versions on a cross-section of device brands and models. For the selection of reference models, a distribution of current Android versions in use was consulted on the Android developers website (Android Open Source Project, 2014a). To summarise, at the time of this writing around 21% of devices run a version of the Gingerbread family (2.3.3–2.3.7), 16% of devices run a variant of the 'Ice Cream Sandwich' family (4.0.3–4.0.4) and over 60% run a variant of the 'Jelly Bean' family (4.1.x–4.3). Table 3 contains an overview of the devices used for the experiments. For each device the brand, model, type and Android version are shown. The Android version is extracted from the field `ro.build.version.release in the` file `/system/build.prop`. An important note has to be made about this file: On rooted devices, it is possible to change values in this file, something that may be done by Android device owners in order to install and run apps that are hard-coded to specific versions or builds of the Android operating system (XDA Developers Forum, 2012). Therefore it is always advisable to check whether this file is modified by looking at file system timestamps and by checking the contents of a similar reference device.

Methodology

A common approach for the reference experiments was followed: first a baseline dump was created, followed by one or more interactions with the device. Next another dump was created and file system metadata and contents of selected files were compared for changes. This is somewhat similar to the approach described by Guido et al. (2013), but the infrastructure described in that article was not available for this research. Instead, full physical dumps after each experiment were created for which a UFED Touch device (Cellebrite, 2014) was used. For each of the experiments that involve the usage of a SIM card, the same SIM card of Dutch Telecom carrier KPN was used in all devices.

The file systems within each dump were analysed using the Sleuth Kit, version 4.1.0 (Carrier, 2014). By comparing fls output and mactime timelines across experiments, relevant files could be identified. SQLite databases were extracted including their corresponding `-shm` and `-wal`, if present. These files are part of SQLite's Write-Ahead Logging mechanism (SQLite Consortium, 2014) and including these files ensures that the current 'allocated' view of the SQLite database was exported. Each SQLite database was transformed to plain text by using the `.dump` command to allow for comparison using standard diff tools. This approach is similar to that described by Vidas et al. (2011). SQLite exports were performed using SQLite version 3.7.17.

Table 3
Reference devices used during the experiments.

Device number	Brand	Model	Type	Android version
1	Sony Ericsson	Xperia	X10i	2.3.3
2	HTC	Sensation	Z710e Pyramid	4.0.3
3	Sony Ericsson	Xperia	X10i	1.6
4	LG	Optimus Black	P970	2.2.2
5	Samsung	Galaxy W	GT-I8150	2.3.6
6	HTC	Endeavor C2	S728e One X+	4.1.1
7	HTC	Desire C	A320e	4.0.3

Time and date related settings & automated time synchronization

As previously stated, it is important to determine whether a device's clock was set to the correct time. This inspired us to perform a more in-depth analysis of the settings in the file `settings.db` located in the path `/data/data/com.android.providers.settings/databases/`, and to examine under which conditions the clock was adjusted to the correct time. For this experiment, the clock was incorrectly set and various settings were adjusted in the devices so as to examine the changes made to the `settings.db` file and the effects of GPS, GSM and Wi-Fi. Apparently, the Android platform was rather late in adopting the use of time synchronisation via NTP,[7] and although it was expected to see the clock adjusted via Wi-Fi in the newer models, it was important to verify this across the broader range of devices and Android versions.

With all devices used in this experiment, the clock was not automatically adjusted via enabling location services and therefore the Android system does not appear to use any GPS provided timestamp to automatically correct the clock. The relevant fields in the `settings.db` in relation to location are `location_providers_allowed` and in some Android versions and implementations `assisted_gps_enabled` is also present. With regards to the Wi-Fi, all of the mobile phones used in the experiment adjusted the clocks almost immediately, even in Android 1.6 which suggests that the time is synchronised via an NTP-server. The relevant field in the `settings.db` is `wifi_on`, and in addition a Wi-Fi network should be configured on the device.

As previously stated in section Automatic time synchronisation, there exists a possibility for date and time to be synchronised via GSM through NITZ. The source

[7] According to a bug report found here: https://code.google.com/p/android/issues/detail?id=18681 (Last Accessed February 2014).

code of Android version 4.0.3_r1 was sought to verify if this was indeed implemented. There are many references to NITZ related code here, but perhaps a comment in the Java class DataBaseHelper.java is most illustrative, since it is also used to initialise settings in the settings.db database. See Fig. 4 for the code fragment. NITZ in general, however, is only supported by a number of Telecom carriers and by certain supported mobile phones. The carrier chosen for our experiments does not support NITZ and therefore during our experimentation no device updated its clock via GSM.

The majority of devices were however updated when Mobile Data was enabled in the Settings of the device. The relevant field in the settings.db is mobile_data. The two Sony Ericsson devices in our reference collection, however, refrained from adjusting the clock even when mobile data and data roaming were enabled. Another strange anomaly was discovered in one of our reference devices: device 2, HTC Sensation Z710e Pyramid. Despite the automatic time and date being disabled as well as disabling of Wi-Fi, so long as Mobile Data was enabled, even when all options in the Mobile Data sub-menu were disabled, the clock was adjusted to the correct time regardless. This reiterates the importance of conducting tests on reference devices of not only the same make and model but also the same Android version during case investigations.

settings.db

An interesting property of the settings.db database was discovered when the reference devices were given a factory reset. As briefly discussed in section Time and date related settings the _id field of settings.db makes use of auto-incrementation and may be exploited for forensic investigation. On the right in Fig. 5 are the settings as seen after a factory reset, whereas the left-hand side of the figure shows the database after a few small changes to the settings.db. Here it is clear to see that as changes are made to a particular setting, a new entry is created in the system or secure table with the same name, but with a different (higher) value in the _id field.

A glance at the code in the Java class DataBaseHelper.java revealed that the records in settings.db are probably initialised in a deterministic manner. This may mean that certain important settings, such as auto_time do not only have some default setting (which happens to be 1, which indicates 'on' on this device), but they may also have a deterministic _id value when first initialised. This fact, in combination with the observation that the _id field is updated on each setting change may provide insight into whether or not a specific setting of interest has been changed since the last factory reset of the device, even if the setting is back at its default value. Further exploration of this interesting observation remains future work. But in the mean time, it is perhaps valuable during a case investigation to reset the reference device to factory defaults so as to compare this file to that of the case in order to determine any changes made to the settings. It should also be noted that some of the settings may also be specific to the particular device in question.

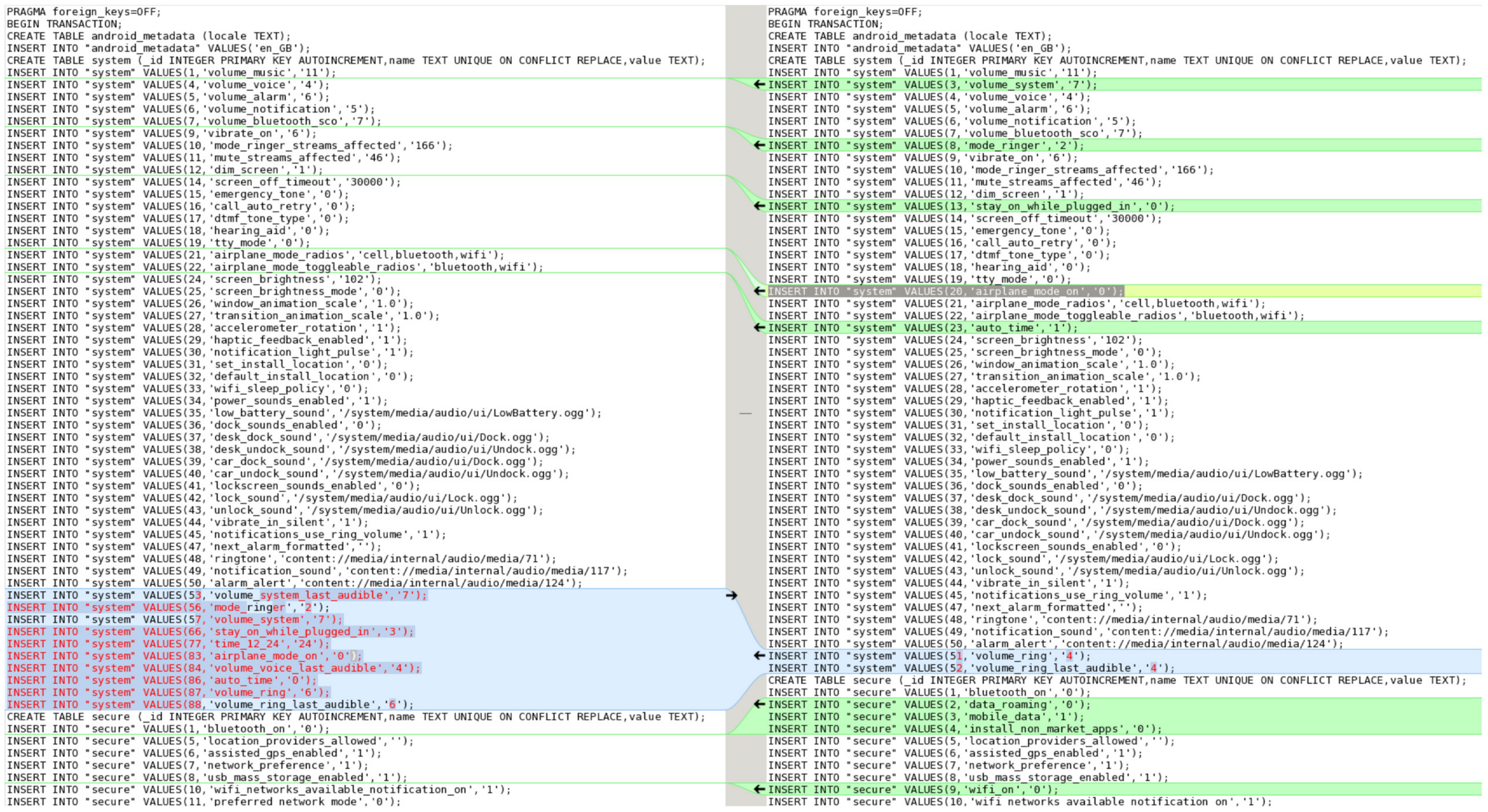
Timestamps in mmssms.db

As previously stated, in one of our cases we exploited the fact that the mobile provider of the device under investigation sent messages about missed calls and voicemail messages that contained a date and timestamp which was set by the providers messaging service. From reference experiments we found that the timestamps in the mmssms.db on this specific Android device were derived from the internal clock of the device. This allowed us to compare both these times and determine whether skew was indeed present. However, in a subsequent case we found that this behaviour was different and the mmssms.db timestamps were not derived from the internal clock.

In order to further understand the significance of the recorded timestamps in the mmssms.db file, an experiment was conducted using the mobile phones listed in Table 3. This experiment consisted of sending SMS messages under two separate conditions for each of the devices. The first condition was where the internal clock was correctly set according to an NTP-server, and the second where the date was incorrectly set. Under the latter conditions, the date and time was prevented from being set correctly by not only disabling the automatic date and time option in the devices settings, but also disabling Wi-Fi and mobile data options. The relevant time field in mmssms.db was then examined.

```
public class DatabaseHelper extends SQLiteOpenHelper {
    private static final String TAG = "SettingsProvider";
    private static final String DATABASE_NAME = "settings.db";
<...>
private void loadSystemSettings(SQLiteDatabase db) {
     SQLiteStatement stmt = null;
     try {
<...>
        loadBooleanSetting(stmt, Settings.System.AUTO_TIME,
               R.bool.def_auto_time); // Sync time to NITZ
        loadBooleanSetting(stmt, Settings.System.AUTO_TIME_ZONE,
               R.bool.def_auto_time_zone); // Sync timezone to NITZ
<...>
```

Fig. 4. Excerpt from the java class DatabaseHelper.java used for initiating settings.db.



```
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE android_metadata (locale TEXT);
INSERT INTO "android_metadata" VALUES('en_GB');
CREATE TABLE system (_id INTEGER PRIMARY KEY AUTOINCREMENT,name TEXT UNIQUE ON CONFLICT REPLACE,value TEXT);
INSERT INTO "system" VALUES(1,'volume_music','11');
INSERT INTO "system" VALUES(4,'volume_voice','4');
INSERT INTO "system" VALUES(5,'volume_alarm','6');
INSERT INTO "system" VALUES(6,'volume_notification','5');
INSERT INTO "system" VALUES(7,'volume_bluetooth_sco','7');
INSERT INTO "system" VALUES(9,'vibrate_on','6');
INSERT INTO "system" VALUES(10,'mode_ringer_streams_affected','166');
INSERT INTO "system" VALUES(11,'mute_streams_affected','46');
INSERT INTO "system" VALUES(12,'dim_screen','1');
INSERT INTO "system" VALUES(14,'screen_off_timeout','30000');
INSERT INTO "system" VALUES(15,'emergency_tone','0');
INSERT INTO "system" VALUES(16,'call_auto_retry','0');
INSERT INTO "system" VALUES(17,'dtmf_tone_type','0');
INSERT INTO "system" VALUES(18,'hearing_aid','0');
INSERT INTO "system" VALUES(19,'tty_mode','0');
INSERT INTO "system" VALUES(21,'airplane_mode_radios','cell,bluetooth,wifi');
INSERT INTO "system" VALUES(22,'airplane_mode_toggleable_radios','bluetooth,wifi');
INSERT INTO "system" VALUES(24,'screen_brightness','102');
INSERT INTO "system" VALUES(25,'screen_brightness_mode','0');
INSERT INTO "system" VALUES(26,'window_animation_scale','1.0');
INSERT INTO "system" VALUES(27,'transition_animation_scale','1.0');
INSERT INTO "system" VALUES(28,'accelerometer_rotation','1');
INSERT INTO "system" VALUES(29,'haptic_feedback_enabled','1');
INSERT INTO "system" VALUES(30,'notification_light_pulse','1');
INSERT INTO "system" VALUES(31,'set_install_location','0');
INSERT INTO "system" VALUES(32,'default_install_location','0');
INSERT INTO "system" VALUES(33,'wifi_sleep_policy','0');
INSERT INTO "system" VALUES(34,'power_sounds_enabled','1');
INSERT INTO "system" VALUES(35,'low_battery_sound','/system/media/audio/ui/LowBattery.ogg');
INSERT INTO "system" VALUES(36,'dock_sounds_enabled','0');
INSERT INTO "system" VALUES(37,'desk_dock_sound','/system/media/audio/ui/Dock.ogg');
INSERT INTO "system" VALUES(38,'desk_undock_sound','/system/media/audio/ui/Undock.ogg');
INSERT INTO "system" VALUES(39,'car_dock_sound','/system/media/audio/ui/Dock.ogg');
INSERT INTO "system" VALUES(40,'car_undock_sound','/system/media/audio/ui/Undock.ogg');
INSERT INTO "system" VALUES(41,'lockscreen_sounds_enabled','0');
INSERT INTO "system" VALUES(42,'lock_sound','/system/media/audio/ui/Lock.ogg');
INSERT INTO "system" VALUES(43,'unlock_sound','/system/media/audio/ui/Unlock.ogg');
INSERT INTO "system" VALUES(44,'vibrate_in_silent','1');
INSERT INTO "system" VALUES(45,'notifications_use_ring_volume','1');
INSERT INTO "system" VALUES(47,'next_alarm_formatted','');
INSERT INTO "system" VALUES(48,'ringtone','content://media/internal/audio/media/71');
INSERT INTO "system" VALUES(49,'notification_sound','content://media/internal/audio/media/117');
INSERT INTO "system" VALUES(50,'alarm_alert','content://media/internal/audio/media/124');
INSERT INTO "system" VALUES(53,'volume_system_last_audible','7');
INSERT INTO "system" VALUES(56,'mode_ringer','2');
INSERT INTO "system" VALUES(57,'volume_system','7');
INSERT INTO "system" VALUES(66,'stay_on_while_plugged_in','3');
INSERT INTO "system" VALUES(77,'time_12_24','24');
INSERT INTO "system" VALUES(83,'airplane_mode_on','0');
INSERT INTO "system" VALUES(84,'volume_voice_last_audible','4');
INSERT INTO "system" VALUES(86,'auto_time','0');
INSERT INTO "system" VALUES(87,'volume_ring','6');
INSERT INTO "system" VALUES(88,'volume_ring_last_audible','6');
CREATE TABLE secure (_id INTEGER PRIMARY KEY AUTOINCREMENT,name TEXT UNIQUE ON CONFLICT REPLACE,value TEXT);
INSERT INTO "secure" VALUES(1,'bluetooth_on','0');
INSERT INTO "secure" VALUES(5,'location_providers_allowed','');
INSERT INTO "secure" VALUES(6,'assisted_gps_enabled','1');
INSERT INTO "secure" VALUES(7,'network_preference','1');
INSERT INTO "secure" VALUES(8,'usb_mass_storage_enabled','1');
INSERT INTO "secure" VALUES(10,'wifi_networks_available_notification_on','1');
INSERT INTO "secure" VALUES(11,'preferred_network_mode','0');
```

```
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE android_metadata (locale TEXT);
INSERT INTO "android_metadata" VALUES('en_GB');
CREATE TABLE system (_id INTEGER PRIMARY KEY AUTOINCREMENT,name TEXT UNIQUE ON CONFLICT REPLACE,value TEXT);
INSERT INTO "system" VALUES(1,'volume_music','11');
INSERT INTO "system" VALUES(3,'volume_system','7');
INSERT INTO "system" VALUES(4,'volume_voice','4');
INSERT INTO "system" VALUES(5,'volume_alarm','6');
INSERT INTO "system" VALUES(6,'volume_notification','5');
INSERT INTO "system" VALUES(7,'volume_bluetooth_sco','7');
INSERT INTO "system" VALUES(8,'mode_ringer','2');
INSERT INTO "system" VALUES(9,'vibrate_on','6');
INSERT INTO "system" VALUES(10,'mode_ringer_streams_affected','166');
INSERT INTO "system" VALUES(11,'mute_streams_affected','46');
INSERT INTO "system" VALUES(12,'dim_screen','1');
INSERT INTO "system" VALUES(13,'stay_on_while_plugged_in','0');
INSERT INTO "system" VALUES(14,'screen_off_timeout','30000');
INSERT INTO "system" VALUES(15,'emergency_tone','0');
INSERT INTO "system" VALUES(16,'call_auto_retry','0');
INSERT INTO "system" VALUES(17,'dtmf_tone_type','0');
INSERT INTO "system" VALUES(18,'hearing_aid','0');
INSERT INTO "system" VALUES(19,'tty_mode','0');
INSERT INTO "system" VALUES(20,'airplane_mode_on','0');
INSERT INTO "system" VALUES(21,'airplane_mode_radios','cell,bluetooth,wifi');
INSERT INTO "system" VALUES(22,'airplane_mode_toggleable_radios','bluetooth,wifi');
INSERT INTO "system" VALUES(23,'auto_time','1');
INSERT INTO "system" VALUES(24,'screen_brightness','102');
INSERT INTO "system" VALUES(25,'screen_brightness_mode','0');
INSERT INTO "system" VALUES(26,'window_animation_scale','1.0');
INSERT INTO "system" VALUES(27,'transition_animation_scale','1.0');
INSERT INTO "system" VALUES(28,'accelerometer_rotation','1');
INSERT INTO "system" VALUES(29,'haptic_feedback_enabled','1');
INSERT INTO "system" VALUES(30,'notification_light_pulse','1');
INSERT INTO "system" VALUES(31,'set_install_location','0');
INSERT INTO "system" VALUES(32,'default_install_location','0');
INSERT INTO "system" VALUES(33,'wifi_sleep_policy','0');
INSERT INTO "system" VALUES(34,'power_sounds_enabled','1');
INSERT INTO "system" VALUES(35,'low_battery_sound','/system/media/audio/ui/LowBattery.ogg');
INSERT INTO "system" VALUES(36,'dock_sounds_enabled','0');
INSERT INTO "system" VALUES(37,'desk_dock_sound','/system/media/audio/ui/Dock.ogg');
INSERT INTO "system" VALUES(38,'desk_undock_sound','/system/media/audio/ui/Undock.ogg');
INSERT INTO "system" VALUES(39,'car_dock_sound','/system/media/audio/ui/Dock.ogg');
INSERT INTO "system" VALUES(40,'car_undock_sound','/system/media/audio/ui/Undock.ogg');
INSERT INTO "system" VALUES(41,'lockscreen_sounds_enabled','0');
INSERT INTO "system" VALUES(42,'lock_sound','/system/media/audio/ui/Lock.ogg');
INSERT INTO "system" VALUES(43,'unlock_sound','/system/media/audio/ui/Unlock.ogg');
INSERT INTO "system" VALUES(44,'vibrate_in_silent','1');
INSERT INTO "system" VALUES(45,'notifications_use_ring_volume','1');
INSERT INTO "system" VALUES(47,'next_alarm_formatted','');
INSERT INTO "system" VALUES(48,'ringtone','content://media/internal/audio/media/71');
INSERT INTO "system" VALUES(49,'notification_sound','content://media/internal/audio/media/117');
INSERT INTO "system" VALUES(50,'alarm_alert','content://media/internal/audio/media/124');
INSERT INTO "system" VALUES(51,'volume_ring','4');
INSERT INTO "system" VALUES(52,'volume_ring_last_audible','4');
CREATE TABLE secure (_id INTEGER PRIMARY KEY AUTOINCREMENT,name TEXT UNIQUE ON CONFLICT REPLACE,value TEXT);
INSERT INTO "secure" VALUES(1,'bluetooth_on','0');
INSERT INTO "secure" VALUES(2,'data_roaming','0');
INSERT INTO "secure" VALUES(3,'mobile_data','1');
INSERT INTO "secure" VALUES(4,'install_non_market_apps','0');
INSERT INTO "secure" VALUES(5,'location_providers_allowed','');
INSERT INTO "secure" VALUES(6,'assisted_gps_enabled','1');
INSERT INTO "secure" VALUES(7,'network_preference','1');
INSERT INTO "secure" VALUES(8,'usb_mass_storage_enabled','1');
INSERT INTO "secure" VALUES(9,'wifi_on','0');
INSERT INTO "secure" VALUES(10,'wifi_networks_available_notification_on','1');
```

Fig. 5. Difference between two versions of the exported `settings.db` on reference device 1.

Table 4
Results of experiment concerning the source of the timestamp as seen in the database file mmssms.db.

Device number	Brand & Model	Android version	Sent message	Received message
1	Sony Ericsson Xperia X10i	2.3.3	internal clock	internal clock
2	HTC Sensation Z710e Pyramid	4.0.3	internal clock	server time
3	Sony Ericsson Xperia X10i	1.6	internal clock	internal clock
4	LG P970	2.2.2	internal clock	internal clock
5	Samsung GT-I8150	2.3.6	internal clock	internal clock
6	HTC Endeavor C2	4.1.1	internal clock	server time
7	HTC Desire C	4.0.3	internal clock	server time

Table 5
Results of the extended experiment to clarify results from first test.

Device number	Brand & Model	Android version	Sent message	Received message
8	HTC Sensation Z710e Pyramid	2.3.3	internal clock	server time
9	Samsung Galaxy Duo	4.0.4	internal clock	internal clock

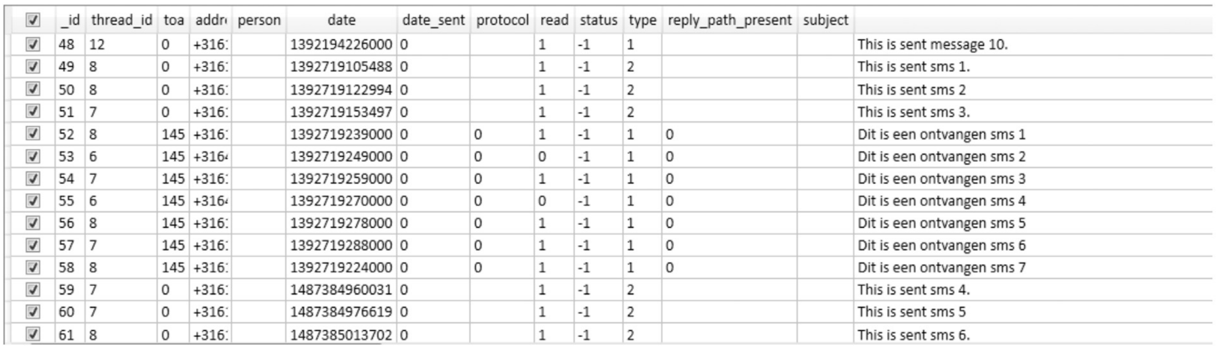
The results of this experiment can be seen in Table 4. It is evident that not all mobile phones used in the experiment utilised the internal clock when recording the SMS in the database. It is feasible that this is dependent on the Android version in question, and that from version 4.x the server time is used for incoming SMS messages. It is also, however, feasible that these results are reliant on the brand and therefore the specific implementation of the Android system. In order to clarify this, another two phones were added to the experiment and tested. So as to ensure that the results of this second test would not be version-reliant, the two added mobile phones were a HTC Explorer running Android version 2.3.3 and a Samsung Galaxy Duo S running version 4.0.4. As seen in Table 5, the results confirmed that the timestamp as written to the file `mmssms.db` was in fact dependant on the brand-specific implementation of the Android system. This behaviour has to date only been observed in HTC mobile phones.

Upon examination of the file `mmssms.db`, it was noted that variations existed between manufacturer-specific implementations of Android. Whereas the `mmssms.db` from most of our devices contained only the date field to store timestamps as seen in Fig. 3, the HTC mobile phones also contained another column called `date_sent` as seen in Fig. 6. However, the value in this field remained 0 in all our experiments. The `mmssms.db` from the LG P970 also contained an extra column entitled `sc_date` which probably contains the time from the SMSC server for received SMS messages as seen in Fig. 7. This gives another opportunity to verify and calculate clock skew. This file should therefore always be checked for a vendor-specific schema.

Furthermore, all physical dumps of the mobile phones used in this experiment were searched for SMS messages still left behind in PDU format. Despite an SMS being deleted from the mobile phone it may be possible to find (fragments of the) PDUs in unallocated space if these were written to the flash memory before unpacking versus remaining in RAM. However PDUs were only present in the physical dump of one device: the HTC Desire C with an Ext4 file system and running Android version 4.0.3. Only approximately about 50% of the received SMS messages were remaining and therefore it is not clear how long these PDUs may remain in the unallocated space. Further research could be done to see if this is another viable source of information (including external SCTS timestamps).

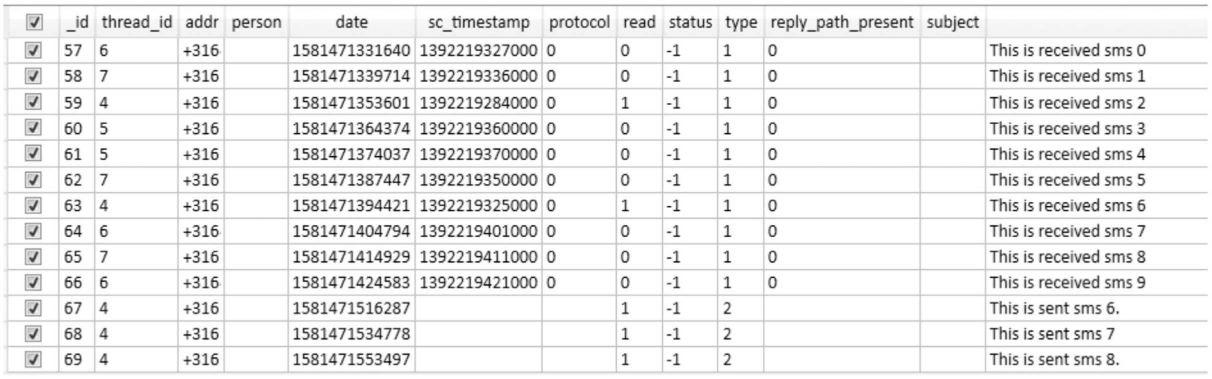
Time zone implementations

The case example in section Case study and the discussion in section Time and date issues clearly show that determining the time zone is important for the correct interpretation of not only the naive timestamps, but also,



✓	_id	thread_id	toa	addr	person	date	date_sent	protocol	read	status	type	reply_path_present	subject	
✓	48	12	0	+316:		1392194226000	0		1	-1	1			This is sent message 10.
✓	49	8	0	+316:		1392719105488	0		1	-1	2			This is sent sms 1.
✓	50	8	0	+316:		1392719122994	0		1	-1	2			This is sent sms 2
✓	51	7	0	+316:		1392719153497	0		1	-1	2			This is sent sms 3.
✓	52	8	145	+316:		1392719239000	0	0	1	-1	1	0		Dit is een ontvangen sms 1
✓	53	6	145	+316		1392719249000	0	0	0	-1	1	0		Dit is een ontvangen sms 2
✓	54	7	145	+316:		1392719259000	0	0	1	-1	1	0		Dit is een ontvangen sms 3
✓	55	6	145	+316		1392719270000	0	0	0	-1	1	0		Dit is een ontvangen sms 4
✓	56	8	145	+316:		1392719278000	0	0	1	-1	1	0		Dit is een ontvangen sms 5
✓	57	7	145	+316:		1392719288000	0	0	1	-1	1	0		Dit is een ontvangen sms 6
✓	58	8	145	+316:		1392719224000	0	0	1	-1	1	0		Dit is een ontvangen sms 7
✓	59	7	0	+316:		1487384960031	0		1	-1	2			This is sent sms 4.
✓	60	7	0	+316:		1487384976619	0		1	-1	2			This is sent sms 5
✓	61	8	0	+316:		1487385013702	0		1	-1	2			This is sent sms 6.

Fig. 6. Excerpt of the `mmssms.db` on one of the HTC devices.



✓	_id	thread_id	addr	person	date	sc_timestamp	protocol	read	status	type	reply_path_present	subject	
✓	57	6	+316		1581471331640	1392219327000	0	0	-1	1	0		This is received sms 0
✓	58	7	+316		1581471339714	1392219336000	0	0	-1	1	0		This is received sms 1
✓	59	4	+316		1581471353601	1392219284000	0	1	-1	1	0		This is received sms 2
✓	60	5	+316		1581471364374	1392219360000	0	0	-1	1	0		This is received sms 3
✓	61	5	+316		1581471374037	1392219370000	0	0	-1	1	0		This is received sms 4
✓	62	7	+316		1581471387447	1392219350000	0	0	-1	1	0		This is received sms 5
✓	63	4	+316		1581471394421	1392219325000	0	1	-1	1	0		This is received sms 6
✓	64	6	+316		1581471404794	1392219401000	0	0	-1	1	0		This is received sms 7
✓	65	7	+316		1581471414929	1392219411000	0	0	-1	1	0		This is received sms 8
✓	66	6	+316		1581471424583	1392219421000	0	0	-1	1	0		This is received sms 9
✓	67	4	+316		1581471516287			1	-1	2			This is sent sms 6.
✓	68	4	+316		1581471534778			1	-1	2			This is sent sms 7
✓	69	4	+316		1581471553497			1	-1	2			This is sent sms 8.

Fig. 7. Excerpt of the `mmssms.db` on the LG device.

when the time zone is set incorrectly, of the UTC timestamps. It also demonstrates the importance of inferring when this setting was last modified. Finally, the example illustrated the importance of knowing how the time zone was implemented on the device under investigation, in order to be able to normalise all the timestamps for further investigation.

For these experiments all network connectivity (Wi-Fi, GSM) was disabled and no SIM card was inserted. Automatic time synchronisation was disabled on all devices. On the devices that supported this option, automatic time zone was disabled as well (devices 6 and 7, HTC Endeavor C2 and HTC Desire C respectively). Then, the time zone was set to Europe/Amsterdam and the device's system clock was set close to the start of the DST period in 2013. It was then observed whether or not DST adjustments were being made. This was repeated for the end of the DST period in 2013 and here too, DST adjustments were observed. Next, the time zone was set to Africa/Casablanca and a similar experiment was performed. Since it was known that the Africa/Casablanca implemented two periods of daylight saving in 2013, this involved four clock adjustments per device.

Results

This experiment showed that on all devices the time zone setting is stored in the same data location: `/data/property/persist.sys.timezone`. As is to be expected, all DST changes for the Europe/Amsterdam time zone were properly followed by each of the devices. For the Africa/Casablanca time zone, however, only one device demonstrated some DST adjustments. On device 6 (HTC Endeavor C2), a DST-adjustment was observed on Sunday the 28th of April at 02:00:00 (local time) when the system clock jumped forward by one hour. It was observed that DST ended on the 29th of September at 03:00:00 (local time) when the system clock jumped back by one hour. These DST transitions do not correspond to the public information that is available for DST transitions in the Africa/Casablanca time zone for 2013 (see Table 1). Instead, they seem to partially correspond to the 2012 DST transitions (DST starts on last Sunday of April and ends on last Sunday of September), but the temporary suspension of DST during the Ramadan religious event (July 20–August 20) (Time and Date AS, 2014) was not observed.

Table 6
Versions of the Time Zone Database on the reference devices.

Device nr.	Brand & Model	Android version	Time Zone Database
1	Sony Ericsson Xperia X10i	2.3.3	2010k
2	HTC Sensation Z710e Pyramid	4.0.3	2011l
3	Sony Ericsson Xperia X10i	1.6	2007h
4	LG P970	2.2.2	2009s
5	Samsung GT-I8150	2.3.6	2011h
6	HTC Endeavor C2	4.1.1	2012c
7	HTC Desire C	4.0.3	2011l

Time Zone Database version

These results prompted a further examination of versions of the Time Zone Database present on the reference devices. Similar to the case study in section Case study, the version of the Time Zone Database was determined for all devices. These versions are shown in Table 6. As can be seen, all devices contain a different version of the Time Zone Database, with the exception of device 2 and 7 (HTC Sensation and HTC Desire C respectively). Interestingly, both devices contain the same version of the Android OS, and it seems likely that the Time Zone Database is determined by the Android distribution release. To verify this, the source code repository for the 4.0.3 branch was examined and here it was found that indeed version 2011l of the Time Zone Database was included in this branch.

In order to verify the observed behaviour for the Africa/Casablanca time zone, the source data for the Time Zone Databases with versions 2012c and the currently available version 2013i were downloaded from IANA's public ftp server and compared. Version 2010k was already available as part of the case study and the relevant part is shown in Fig. 8. The lines starting with "Rule" contain DST rules for the given period, which is delimited by the FROM and TO field. When the TO field contains the term 'only', the RULE only applies to that year. As can be seen, there are no additional rules for DST present, which means that no DST

```
# RULE   NAME     FROM    TO      TYPE    IN      ON      AT      SAVE    LETTER/S
<..>
Rule     Morocco  1974    only    -       Jun     24       0:00   1:00    S
Rule     Morocco  1974    only    -       Sep      1       0:00   0       -
Rule     Morocco  1976    1977    -       May      1       0:00   1:00    S
Rule     Morocco  1976    only    -       Aug      1       0:00   0       -
Rule     Morocco  1977    only    -       Sep     28       0:00   0       -
Rule     Morocco  1978    only    -       Jun      1       0:00   1:00    S
Rule     Morocco  1978    only    -       Aug      4       0:00   0       -
Rule     Morocco  2008    only    -       Jun      1       0:00   1:00    S
Rule     Morocco  2008    only    -       Sep      1       0:00   0       -
Rule     Morocco  2009    only    -       Jun      1       0:00   1:00    S
Rule     Morocco  2009    only    -       Aug      21      0:00   0       -
Rule     Morocco  2010    only    -       May      2       0:00   1:00    S
Rule     Morocco  2010    only    -       Aug      8       0:00   0       -
# Zone   NAME             GMTOFF  RULES   FORMAT  [UNTIL]
Zone  Africa/Casablanca   -0:30:20 -      LMT     1913 Oct 26
                           0:00   Morocco WE%sT   1984 Mar 16
                           1:00   -       CET     1986
                           0:00   Morocco WE%sT
```

Fig. 8. Rules in the 2010k version of the Time Zone Database pertaining to the Africa/Casablanca time zone.

```
# RULE   NAME     FROM    TO      TYPE    IN      ON      AT      SAVE    LETTER/S
<...>
Rule     Morocco  2010    only    -       May      2       0:00   1:00    S
Rule     Morocco  2010    only    -       Aug      8       0:00   0       -
Rule     Morocco  2011    only    -       Apr      3       0:00   1:00    S
Rule     Morocco  2011    only    -       Jul      31      0      0       -
Rule     Morocco  2012    max     -       Apr      lastSun 2:00   1:00    S
Rule     Morocco  2012    max     -       Sep      lastSun 3:00   0       -
# Zone   NAME             GMTOFF  RULES   FORMAT  [UNTIL]
Zone  Africa/Casablanca   -0:30:20 -      LMT     1913 Oct 26
                           0:00   Morocco WE%sT   1984 Mar 16
                           1:00   -       CET     1986
                           0:00   Morocco WE%sT
```

Fig. 9. Rules in the 2012c version of the Time Zone Database pertaining to the Africa/Casablanca time zone.

is enforced after 2010 on systems with this version of the Time Zone Database. Fig. 9 contains the relevant rules for the 2012c version of the database and as can be seen, the rules correspond to the observed behaviour on device 6 (HTC Endeavor C2). Fig. 10 contains an extract from the rules for this time zone as present in the current 2013i version of the Time Zone Database. As can be seen, here the DST rules correspond to those represented in Table 1.

Discussion

The work presented here is focused on interpretation of timestamps encountered on Android devices. Each of the topics covered here can be addressed in more detail.

For example, while the research concerning automatic time synchronisation revealed some interesting anomalies, the possibilities for additional research are seemingly endless. In fact, the study of automatic time synchronisation on Android and other mobile devices would justify an entire article in itself.

For example, it is possible to capture network traffic and observe whether NTP traffic occurs. Also, SIM cards from different mobile carriers can be tested in order to verify synchronisation based on NITZ and its accuracy. Also, in our experiments clock skew of several years was introduced, while it is possible that under more common conditions time synchronisations would occur less frequently or not at all, a possibility suggested in one of the issues on the issue tracker page for the Android project.[8]

The importance of understanding time zone issues has been presented, as well as the importance for keeping an eye open for such details. It would be a nice addition to the field when timestamp normalisation performed by common tools could be based on the Time Zone Database encountered on the device under investigation. Instead, we suspect that tools such as UFED, FTK, EnCase and The Sleuth Kit perform timestamp normalisation based on the Time Zone Database available on the analysis computer, something that may be worth investigating further. In addition, the case study incidentally involved a somewhat dynamic time zone in the sense that on the political level some recent changes were introduced.[9] Perhaps the problem of Time Zone Database versioning is not so pressing in most other cases.

Something similar holds for the interpretation of timestamps in the `mmssms.db` database and for the settings in the `settings.db` database. Both these topics could have been explored in much greater detail. The clock skew

[8] See: http://code.google.com/p/android/issues/detail?id=12476 (Last Accessed February 2014).

[9] For example, the timeanddate.com website states the following: “Morocco will switch to daylight saving time (DST) every year from 2012 onwards. The North African country had revived DST as early as 2008, but the new law passed by the Moroccan government council on March 8 formalizes the seasonal time change for years to come.”. See: http://www.timeanddate.com/news/time/morocco-dst-2012.html (Last Accessed February 2014).

```
# RULE  NAME     FROM    TO      TYPE    IN      ON      AT      SAVE    LETTER/S
<...>
Rule    Morocco  2012    2013    -       Apr     lastSun 2:00    1:00    S
Rule    Morocco  2012    only    -       Sep     30      3:00    0       -
Rule    Morocco  2012    only    -       Jul     20      3:00    0       -
Rule    Morocco  2012    only    -       Aug     20      2:00    1:00    S
Rule    Morocco  2013    only    -       Jul      7      3:00    0       -
Rule    Morocco  2013    only    -       Aug     10      2:00    1:00    S
Rule    Morocco  2013    2035    -       Oct     lastSun 3:00    0       -
Rule    Morocco  2014    2022    -       Mar     lastSun 2:00    1:00    S
Rule    Morocco  2014    only    -       Jun     29      3:00    0       -
Rule    Morocco  2014    only    -       Jul     29      2:00    1:00    S
<...>
# Zone  NAME             GMTOFF  RULES   FORMAT  [UNTIL]
Zone  Africa/Casablanca  -0:30:20 -      LMT     1913 Oct 26
                          0:00   Morocco WE%sT   1984 Mar 16
                          1:00   -       CET     1986
                          0:00   Morocco WE%sT
```

Fig. 10. Rules in the 2013i version of the Time Zone Database pertaining to the Africa/Casablanca time zone.

detection method presented in the case study in section Case study can only be used when external timestamps are present, either within the body text or when both device and SMS Service Centre timestamps are present in the databases.

Conclusions and future work

The case study presented here clearly demonstrates the importance of paying attention to details when it comes to time and date issues. The main conclusion of this part of the work is that it remains important to verify the behaviour on a reference device of the same brand and model and with the same Android version, prior to making any conclusions on timestamps on the device under investigation.

Some relevant data locations and settings related to date and time were explored and a relatively simple method for detection of clock skew was introduced. As far as automatic time synchronisation is concerned: the settings of influence have been identified, but the exact conditions under which time synchronisation takes place could be the subject of future work. For case work, this kind of behaviour should always be examined on a reference device.

The work also demonstrated that the Time Zone Database versions present in the Android distribution are lagging behind the currently released version of the database. But even when a more recent version of the Time Zone Database is installed on a device, this is no guarantee for an accurate representation of actual time zone rules in place. One of the comments in the source files downloaded from IANA's ftp server states that 'these files are by no means authoritative'. This suggests that differences between actual time zone rules and those implemented on devices will always remain an issue. Verifying the version of the Time Zone Database and comparing the rules that apply to the configured time zone with those recorded in the most recent version of the Time Zone Database would be a good addition to the checklist proposed by Boyd and Forster (2004). A possible topic for future work would be to create tools that can perform timestamp normalisation based on the Time Zone Database present on the device under investigation.

Finally, an interesting property of the database `settings.db` was observed. The auto-incrementing `_id` field can potentially be used to identify whether a specific setting in the database has ever been modified since the last factory reset. This too remains a possible area of future research. It is likely that such behaviour is also exhibited on other SQLite databases.

In general, it is desirable that more research will be presented that is focused more on the analytical aspect of Android Forensics as this area seems underexposed.

References

Android Open Source Project. Dashboards – Android Developers website. Online, http://developer.android.com/about/dashboards/index.html; 2014a [Last Accessed February 2014].

Android Open Source Project. Downloading the Source – Android Developers website. Online, http://source.android.com/source/downloading.html; 2014b [Last Accessed February 2014].

Android Open Source Project. DropBoxManager – Android Developers website. Online, http://https://developer.android.com/reference/android/os/DropBoxManager.html; 2014c [Last Accessed February 2014].

Barmpatsalou K, Damopoulos D, Kambourakis G, Katos V. A critical review of 7 years of mobile device forensics. Digit Investig 2013;10(4):323–49.

Boyd C, Forster P. Time and date issues in forensic computing-a case study. Digit Investig 2004;1(1):18–23.

Buchholz F, Tjaden B. A brief study of time. Digit Investig 2007;4(Suppl. 0):31–42.

Carrier B. File system forensic analysis. Addison-Wesley Professional; 2005.

Carrier B. The Sleuth Kit (TSK). Online, http://www.sleuthkit.org; 2014 [Last Accessed February 2014].

Cellebrite. UFED touch. Online, http://www.cellebrite.com/mobile-forensics/products/standalone/ufed-touch-ultimat; 2014 [Last Accessed February 2014].

Chung H, Park J, Lee S, Kang C. Digital forensic investigation of cloud storage services. Digit Investig 2012;9(2):81–95.

ETSI. Network Identity and TimeZone (NITZ); Service description; Stage 1 (3GPP TS 22.042 version 11.0.0 Release 11). Online, http://www.3gpp.org/DynaReport/22042.htm; 1998a [Last Accessed February 2014].

ETSI. Technical realization of the Short Message Service (SMS) (3GPP TS 03.40 version 7.5.0 Release 1998). Online, http://www.3gpp.org/DynaReport/0340.htm; 1998b [Last Accessed February 2014].

Fairbanks KD. An analysis of ext4 for digital forensics. Digit Investig 2012; 9(Suppl. 0):S118–30.

Guido M, Ondricek J, Grover J, Wilburn D, Nguyen T, Hunt A. Automated identification of installed malicious Android applications. Digit Investig 2013;10(Suppl. 0):S96–104.

Hoog A. Android forensics: investigation, analysis and mobile security for Google Android. 1st ed. Syngress Publishing; 2011.

IANA. IANA – Time Zone Database. Online, https://www.iana.org/time-zones; 2014 [Last Accessed February 2014].

Lessard J, Kessler GC. Android forensics: simplifying cell phone examinations. Small Scale Digit Device Forensics J 2010;4(1).

Linux Kernel Organization, Inc.. Linux kernel documentation – vfat.txt. Online, https://www.kernel.org/doc/Documentation/filesystems/vfat.txt; 2013 [Last Accessed February 2014].

Manning C. How Yaffs works. Online, http://www.yaffs.net/documents/how-yaffs-works; 2012 [Last Accessed February 2014].

Mills D, Delaware U, Martin J, Burbank J, Kash W. RFC 5905-Network Time Protocol version 4: protocol and algorithms specification. Online, http://www.ietf.org/rfc/rfc5905.txt; 2010 [Last Accessed February 2014].

Minnaard W. Retroactively estimating system clock skew from stored web browser cookies. Online, http://nontrivialpursuit.org/cookieskewer/retroestimate.pdf; 2013 [Last Accessed February 2014].

Mutawa NA, Baggili I, Marrington A. Forensic analysis of social networking applications on mobile devices. Digit Investig 2012; 9(Suppl. 0):S24–33.

Quick D, Alzaabi M. Forensic analysis of the Android file system Yaffs2. In: Proceedings of The 9th Australian Digital Forensics Conference. Western Australia: Security Research Centre Edith Cowan University Perth; 2011. pp. 100–9.

Racioppo C, Murthy N. Android forensics: a case study of the "HTC Incredible" phone. In: Proceedings of Student-Faculty Research Day. CSIS, Pace University. Seidenberg School of CSIS, Pace University, New York; 2012. B6.1–8.

Simao A, Sicoli F, Melo L, Deus F, de Sousa R. Acquisition and analysis of digital evidence in Android smartphones. Int J FORENSIC Comput Sci 2011;1:28–43.

SQLite Consortium. SQLite – Write-Ahead Logging. Online, https://www.sqlite.org/draft/wal.html; 2014 [Last Accessed February 2014].

Sylve J, Case A, Marziale L, Richard GG. Acquisition and analysis of volatile memory from android devices. Digit Investig 2012;8(3–4):175–84.

Thing VLL, Ng KY, Chang EC. Live memory forensics of mobile phones. Digit Investig 2010;7:S74–82.

Time and Date AS. timeanddate.com website. Online, http://www.timeanddate.com; 2014 [Last Accessed February 2014].

Vidas T, Zhang C, Christin N. Toward a general collection methodology for Android devices. Digit Investig 2011;8:S14–24.

XDA Developers Forum. How to edit build.prop on any Android device!. Online, http://forum.xda-developers.com/showthread.php?t=1948558; 2012 [Last Accessed February 2014].