Forensic analysis of WhatsApp Messenger on Android smartphones

Cosimo Anglano*

DiSIT – Computer Science Institute, Universitá del Piemonte Orientale, Alessandria, Italy

ARTICLE INFO

Article history:
Received 15 January 2014
Received in revised form 17 March 2014
Accepted 26 April 2014

Keywords:
WhatsApp
WhatsApp Messenger
Instant messaging
Android
Smartphones

ABSTRACT

We present the forensic analysis of the artifacts left on Android devices by *WhatsApp Messenger*, the client of the WhatsApp instant messaging system. We provide a complete description of all the artifacts generated by WhatsApp Messenger, we discuss the decoding and the interpretation of each one of them, and we show how they can be correlated together to infer various types of information that cannot be obtained by considering each one of them in isolation.

By using the results discussed in this paper, an analyst will be able to reconstruct the list of contacts and the chronology of the messages that have been exchanged by users. Furthermore, thanks to the correlation of multiple artifacts, (s)he will be able to infer information like when a specific contact has been added, to recover deleted contacts and their time of deletion, to determine which messages have been deleted, when these messages have been exchanged, and the users that exchanged them.

Introduction

The introduction of sophisticated communication services over the Internet, allowing users to exchange textual messages, as well as audio, video, and image files, has changed the way people interact among them. The usage of these services, broadly named *instant messaging (IM)*, has undoubtedly exploded in the past few years, mainly thanks to the pervasiveness of smartphones, that provide quite sophisticated IM applications. Smartphones indeed enable users to exploit their data connection to access IM services anywhere and anytime, thus eliminating the costs usually charged by mobile operators for similar services (e.g., for SMS communication).

Given their popularity, IM services are being increasingly used not only for legitimate activities, but also for illicit ones (The United Nations Office on Drugs and Crime, 2013): criminals may indeed use them either to communicate with potential victims, or with other criminals to escape interception (Bellovin et al., 2013). Therefore, IM applications have the potential of being a very rich source of evidentiary information in most investigations.

Among IM applications for smartphones, *WhatsApp* (WhatsApp Inc., 2013) is accredited to be the most widespread one (reportedly (Winkler, 2013), it has over 400 million active users that exchange, on average, more than 31 billion messages per day, 325 millions of which are photos (Olivarez-Giles, 2013)). Given its recent acquisition by Facebook, it is reasonable to expect a further growth in its diffusion. Therefore, the analysis of *WhatsApp Messenger*, the client of WhatsApp that runs on smartphones, has recently raised the interest of the digital forensics community (Thakur, 2013; Mahajan et al., 2013; Tso et al., 2012).

In this paper we deal with the forensic analysis of WhatsApp Messenger on Android smartphones. Android users, indeed, arguably represent the largest part of the

* Tel.: +39 0131 360188.
E-mail addresses: cosimo.anglano@unipmn.it, cosimo.anglano@di.unipmn.it.

http://dx.doi.org/10.1016/j.diin.2014.04.003

user base of WhatsApp: as of Jan. 2014, Google Playstore reports a number of downloads included between 100 and 500 millions (the lower limit having been already hit in Nov. 2012), out of a population of 400 millions of users. Thus, by focusing on the Android platform, we maximize the potential investigative impact of our work.

Several works, appeared recently in the literature (Thakur, 2013; Mahajan et al., 2013), deal with the same problem. However, as discussed later, these works are limited in scope, as they focus only on the reconstruction of the chronology of exchanged messages, and neglect other important artifacts that, instead, are considered in our work.

More precisely, the contributions of this paper can be summarized as follows:

- We discuss the decoding and the interpretation of all the artifacts and data generated by WhatsApp Messenger on Android devices;
- We show how these artifacts can be correlated together to infer various types of information that cannot be obtained by considering each one of them in isolation, such as when a contact has been added to or deleted from the contacts database, whether a message has been actually delivered to its destination after having been sent or has been deleted, if a user joined or left a group chat before or after a given time, when a given user has been added to the list of contacts, etc.

The rest of the paper is organized as follows. In Sec. Related works we review existing work, while in Sec. Analysis methodology and tools we describe the methodology and the tools we use in our study. Then, in Sec. Forensic analysis of WhatsApp Messenger we discuss the forensic analysis of WhatsApp Messenger and, finally, in Sec. Conclusions we conclude the paper and outline future research work.

Related works

The forensic analysis of IM applications on smartphones has been the subject of various works published in the literature.

Compared with existing works, however, our contribution (a) has a wider scope, as it considers all the artifacts generated by WhatsApp Messenger (namely, the database of contacts, the log files, the avatar pictures, and the preference files), (b) presents a more thorough and complete analysis of these artifacts, and (c) explains how these artifacts can be correlated to deduce various type of information having an evidentiary value, such as whether a message has been actually delivered to its destination after having been sent, if a user joined or left a group chat before or after a given time, and when a given user has been added to the list of contacts.

(Husain and Sridhar, 2010) focus on the forensic analysis of three IM applications (namely AIM, Yahoo! Messenger, and Google Talk) on the iOS platform. Their work differs from ours for both the IM applications and the smartphone platform it considers.

(Barghuthi and Said, 2013) focus on the analysis of several IM applications (including WhatsApp Messenger) on various smartphone platforms, including Android, with the aim of identifying the encryption algorithms used by them. Their work, unlike ours, does not deal with the identification, analysis, and correlation of all the artifacts generated by WhatsApp Messenger.

(Tso et al., 2012) focus on the analysis of iTunes backups for iOS devices with the aim of identifying the artifacts left by various social network applications, including WhatsApp Messenger. Their work differs from ours because of its focus on iTunes and iOS, and because just the chat database of WhatsApp is considered, since only this artifact is included into an iTune backup. Furthermore, the information stored into the chat database is analyzed only in part.

The works of (Thakur, 2013) and (Mahajan et al., 2013) are similar to ours, since they both focus on the forensic analysis of WhatsApp Messenger on Android. However, these works focus mainly on the forensic acquisition of the artifacts left by WhatsApp Messenger, and deal with their analysis only in part (they limit their study to the chat database, and analyze it only partially). Similar considerations apply to the *WhatsApp Xtract* tool (Sangiacomo and Weidner, 2012), that extracts some of the information stored into the chat database (and, possibly, in the contacts database), without however providing any description of how these databases are parsed.

Analysis methodology and tools

The study described in this paper has been performed by carrying out a set of controlled experiments, each one referring to a specific usage scenario (one-to-one communication, group communication, multimedia message exchange, etc.), during which typical user interactions have taken place. After each experiment, the memory of the sending and receiving devices has been examined in order to identify, extract, and analyze the data generated by WhatsApp Messenger in that experiment.

As discussed in Sec. Forensic analysis of WhatsApp Messenger, most of the files generated by WhatsApp Messenger are stored into an area of the internal device memory that is normally inaccessible to users. To access this area, suitable commercial tools may be used (Cellebrite LTD., 2013; Micro Systemation, 2013; Oxygen Forensics, Inc., 2013a) but, unfortunately, we did not have access to them. Open-source software-based tools are also available (Hoog, 2011; Vidas et al., 2011), but we consider problematic their use in our study for the following reasons:

- they may alter the contents of the memory, thus overwriting pieces of information: while this can be considered acceptable in a real-world investigation when there is no other alternative, we believe that it should be avoided in a study like the one presented in this paper, as modified or incomplete data may yield to incorrect conclusions;
- they are device-specific: this would prevent a third party to replicate the experiments to validate our findings, unless the same device model and the same software acquisition tool are used.

For these reasons, in our study we adopted a different approach, in which we use software-emulated Android devices in place of physical ones. In particular, we use the

YouWave virtualization platform (YouWave Corp., 2013) that is able to faithfully emulate the behavior of a complete Android device. YouWave implements the internal device memory as a VirtualBox storage file (Oracle Corp., 2013), whose format is documented and, therefore, can be parsed by a suitable tool to extract the files stored inside it. In this way, the acquisition of the internal memory of the device is greatly simplified, as it reduces to inspect the content of this file.

In order to ensure the soundness of our approach, we have made tests in which the behavior of, and the data generated by, WhatsApp Messenger running on YouWave have been compared against those produced when it runs on real smartphones. These tests have been performed either indirectly, by comparing the data found in the inaccessible memory area of YouWave against those documented in the literature (Thakur, 2013; Mahajan et al., 2013), or directly, by comparing the data stored on the emulated SD memory card against those generated on a real smartphone. The results of our tests indicate that, from the perspective of WhatsApp Messenger, YouWave and a real smartphone behave the same way.

Our experimental test-bed consists thus into a set of YouWave virtual machines, namely one for each device involved in the experiments, running Android v. 4.0.4. On each one of these machines we install and use WhatsApp Messenger v. 2.11. In each experiment, we assign a role to each virtual device (e.g. sender or recipient of a message, group chat leader, etc.), and use it to carry out the corresponding activities. Then, at the end of the experiment, we suspend the virtual device, parse the file implementing the corresponding internal memory (named `youwave_vm01.vdi`) by means of *FTK Imager* (v. 3.1) (AccessData Corporation, 2013), and extract the files where WhatsApp Messenger stores the data it generates.[1] These files are then examined by means of suitable tools. In particular, we use *SqliteMan* (Vanek and Les, 2013) to examine the databases maintained by WhatsApp Messenger (as discussed later, they are SQLite v.3 databases (SQLite Consortium, 2013)), and `notepad++` (Ho, 2013) to examine textual files.

By proceeding as exposed above, (a) we are able to avoid the risks of contamination and of an incomplete acquisition of the data stored in the memory of the device, (b) we ensure repeatability of experiments, as their outcomes do not depend on the availability of a specific software or hardware memory acquisition tool or smartphone model, (c) we obtain a high degree of controllability of experiments, as we may suspend and resume at will the virtual device to perform acquisition while a given experiment is being carried out and, last but not less important, (d) we reduce the costs of the study, since neither real smartphones nor commercial memory acquisition tools are necessary to carry out the experiments.

[1] The only exception we make to the above methodology is the use of a physical smartphone to generate messages carrying geolocation coordinates, since the Android Location Services, used by WhatsApp Messenger to obtain the coordinates of the current location of the device, are not available on YouWave because of its lack of a GPS receiver. In this case, access to the relevant data is achieved by using the backup mechanisms described in Sec. The structure of the chat database.

Forensic analysis of WhatsApp Messenger

WhatsApp provides its users with various forms of communications, namely user-to-user communications, broadcast messages, and group chats. When communicating, users may exchange plain text messages, as well as multimedia files (containing images, audio, and video), contact cards, and geolocation information.

Each user is associated with a *profile*, a set of information that includes his/her *WhatsApp name*, *status line*, and *avatar* (a graphic file, typically a picture). The profile of each user is stored on a central system, from which it is downloaded by other WhatsApp users that include that user in their contacts. The central systems provides also other services, like user registration, authentication, and message relay.

As reported in (Thakur, 2013), the artifacts generated by WhatsApp Messenger on an Android device are stored into a set of files, whose name, location, and contents are listed in Table 1.

In the rest of this section we discuss how the above artifacts can be analyzed and correlated to ascertain various types of information: we start with contact information (Sec. Analysis of contact information), we continue with exchanged messages (Sec. Analysis of exchanged messages), and we end with application settings and user preferences (Sec. Analysis of settings and preferences).

Analysis of contact information

The evidentiary value of contact information is notorious, as it allows an investigator to determine who the user was in contact with.

In this section we first describe the information that are stored in the contacts database, and then we discuss how this information can be analyzed to determine (a) the list of contacts, (b) when a contact has been added to the database, (c) whether and when a given contact has been blocked and, finally, we show how deleted contacts can be dealt with.

Retrieving contact information

The contacts database `wa.db` contains three tables, namely `wa_contacts`, that stores a record for each contact, `android_metadata`, and `sqlite_sequence`, both storing housekeeping information having no evidentiary value.

The structure of the records in `wa_contacts` is shown in Table 2, where we distinguish the fields containing data obtained from the WhatsApp system (and, as such, having potential evidentiary value), from those storing data extracted from the phonebook of the device (that, being set by the user and not by WhatsApp, are not pertinent to our work).

As can be observed from this table, each record stores the WhatsApp ID (field `jid`) of the contact, a string structured as '`x@s.whatsapp.net`', where '`x`' is the phone number of that contact (for the sake of readability, in the following we indicate users by means of their phone numbers instead of their complete WhatsApp IDs). Furthermore, each record stores the profile name (field

Table 1
WhatsApp Messenger artifacts.

Row#	Content	Directory	File
1	contacts database	/data/data/com.whatsapp/databases	wa.db (SQLite v.3)
2	chat database	/data/data/com.whatsapp/databases	msgstore.db (SQLite v.3)
3	backups of the chat database	/mnt/sdcard/Whatsapp/Databases	msgstore.db.cryptmsgstore-< date>.crypt
4	avatars of contacts	/data/data/com.whatsapp/files/Avatars	UID.j, where UID is the identifier of the contact
5	copies of contacts avatars	/mnt/sdcard/WhatsApp/ProfilePictures	UID.j, where UID is the identifier of the contact
6	log files	/data/data/com.whatsapp/files/Logs	whatsapp.log, whatsapp-< date>.log
7	received files	/mnt/sdcard/Whatsapp/Media	various files
8	sent files	/mnt/sdcard/Whatsapp/Media/Sent	various files
9	user settings and preferences	/data/data/comm.whatsapp/files	various files

wa_name), and the status string (field status) of the corresponding contact. Field is_whatsapp_user is instead used to differentiate actual WhatsApp users from unreal ones: WhatsApp Messenger indeed adds to the contact database a record for each phone number found in the phonebook of the device, even if the corresponding user is not registered with the WhatsApp system.

Avatar pictures may have evidentiary value as well: they can be indeed used to link a WhatsApp account to the real identity of the person using it (for instance, if the avatar displays the face of the user, or any location or item that can be uniquely associated with that person). The avatar picture of a contact x@s.whatsapp.net is stored, as a JPEG file named x@s.whatsapp.net.j, in the directories listed in Table 1, rows no. 4 and 5. The timestamps stored in the thumb_ts and photo_id_timestamp field indicate when the contacts has set his/her current avatar, and when that avatar has been downloaded locally, respectively.

Table 2
Structure of the *wa_contacts* table.

Field name	Meaning
Data coming from the WhatsApp system	
_id	sequence number of the record (set by SQLite)
jid	WhatsApp ID of the contact (a string structured as 'x@s.whatsapp.net', where 'x' is the phone number of the contact)
is_whatsapp_user	contains '1' if the contact corresponds to an actual WhatsApp user, '0' otherwise
unseen_msg_count	number of messages sent by this contact that have been received, but still have to be read
photo_ts	unknown, always set to '0'
thumb_ts	Unix epoch time (10 digits) indicating when the contact has set his/her current avatar picture
photo_id_timestamp	Unix millisecond epoch time (13 digits) indicating when the current avatar picture of the contact has been downloaded locally
wa_name	WhatsApp name of the contact (as set in his/her profile)
status	status line of the contact (as set in his/her profile)
sort_name	name of the contact used in sorting operations
Data coming from from the phonebook of the device	
number	phone number associated to the contact
raw_contact_id	sequence number of the contact
display_name	display name of the contact
phone_type	type of the phone
phone_label	label associated to the phone number
given_name	given name of the user
family_name	family name of the user

Determining when a contact has been added

In some investigations, it may be necessary to determine when a given user has been added to the contacts database.[2] This information is not stored in the wa_contacts table, but can be deduced from the analysis of the log files generated by WhatsApp Messenger (that are located in the directory listed in Table 1, row no. 6).

When a contact is added to the wa.db database, WhatsApp Messenger logs several events that are tagged with their time of occurrence and with the WhatsApp ID of the involved user.

Examples of these events, corresponding to the addition of user *39331xxxxxxx*, are reported in Fig. 1, from which we note that the following events are logged each time a new user is added: (a) the discovery that the user is not present yet in the contacts database (line no. 4), (b) the queries to the central system to fetch various information about the contact (lines no. 7,10, and 14), and (c) the completion of the download of the corresponding avatar picture (line no. 17). From these events, we can determine when the user has been added to the contacts database (on Sept. 25, 2013 at 14:14:24, in our example).

Dealing with blocked contacts

WhatsApp Messenger enables the user to block anyone of his/her contacts, thus preventing any communication with him/her until the block is removed. In an investigation it can be important to determine whether a contact was blocked or not at a given time, in order to confirm or to exclude the reception of a message sent at that time.

The information concerning blocked users is stored neither in the contacts database, nor elsewhere on the memory of the device (we conjecture that the list of

[2] User contacts are automatically added to the contacts database by WhatsApp Messenger that – each time is started or when the user starts a new conversation – inspects the phonebook of the device and adds all the. phone numbers that are not stored there yet.

Please cite this article in press as: Anglano C, Forensic analysis of WhatsApp Messenger on Android smartphones, Digital Investigation (2014), http://dx.doi.org/10.1016/j.diin.2014.04.003

```
2013-09-25 14:14:24.161 I: [1] contactpicker/create
2013-09-25 14:14:24.162 I: [1] 1 contacts selected for picker
                               (is_broadcast=false) | time: 1
2013-09-25 14:14:24.201 I: [89] found 0 similar contacts to row_id=1
                                jid=39331XXXXXXX@s.whatsapp.net
                                key=1-331XXXXXXX phone=2 iswa=true | time: 0
2013-09-25 14:14:24.201 I: [89] app/sendGetProfilePhoto photo_id:0 type:2
                                jid:39331XXXXXXX@s.whatsapp.net
[...]
2013-09-25 14:14:24.343 I: [82] xmpp/reader/read/profilephotoreceived
                                39331XXXXXXX@s.whatsapp.net id:1363544071
                                type:preview has_data:true
[...]
2013-09-25 14:14:24.344 I: [1] contact fetched by jid=39331XXXXXXX@s.whatsapp.net
                               result=row_id=1 jid=39331XXXXXXX@s.whatsapp.net
                               key=1-331XXXXXXX phone=2 iswa=true count=1 | time: 1
2013-09-25 14:14:24.364 I: [67] updated photo id for contact
                                jid=39331XXXXXXX@s.whatsapp.net
                                photo_id_timestamp=1380118464344 thumb_ts=1363544071
                                photo_ts=0 | time: 20
```

Fig. 1. Events logged when a user is added to the `contacts` table (phonenumber redacted to ensure the privacy of the owner). Long lines have been split to ease readability.

blocked contacts is stored on the WhatsApp central system, since when the blocking is taking place, WhatsApp Messenger exchanges messages with it). Blocked users can be however identified, under some circumstances, by analyzing log files.

When a contact is blocked, an event – reporting the WhatsApp ID of that contact and the time of occurrence of the operation – is indeed recorded into the log file (see Fig. 2(a)). Unfortunately, when a contact is unblocked, the event that is logged (Fig. 2(b)) does not report the WhatsApp IDs of the involved contact, and it is cumulative (i.e., it may refer to a set of contacts being unblocked simultaneously).

Thus, it is always possible to determine whether and when a given user `x` has been blocked, but whether it is still blocked at a given time can be ascertained only if either (a) no unblocking events are recorded in the log file after the block operation, or (b) an unblocking event is present, but only user `x` was blocked at that time. It follows that if several users are blocked and one (or more) unblocking events are logged, it is not possible to tell which users are still blocked, and which ones have been instead unblocked. It is worth pointing out that the above inferences can be made only if the log files reporting blocking and unblocking events are still available (i.e., they have not been deleted by WhatsApp Messenger to create room for newer ones).

As a final consideration, we note that no information whatsoever is stored on the side of the contact that gets blocked, so it is not possible to tell whether the user of the device under analysis has been blocked or not by anyone of his/her contacts.

Dealing with deleted contacts

In the attempt to hide past interactions, the user may delete a contact, thus causing the removal of the corresponding record from the `wa_contacts` table.

In some cases (notably, if the SQLite engine has not vacuumed the above table yet), it may be possible to recover deleted records by means of suitable techniques (e.g., (Jeon et al., 2012; Pooters et al., 2011)). Our experiments, carried out by means of Oxygen Forensic SQLite Viewer (Oxygen Forensics, Inc., 2013b), indicate indeed that deleted contact records may be recovered.

However, in general, at the moment of the analysis, deleted records may have been vacuumed, so they cannot be recovered anymore. In these situations, it may be still possible to determine the set of deleted contacts by first reconstructing the list of contacts that have been added in the past (by analyzing log files as discussed in Sec. Determining when a contact has been added), and then by comparing this list with the contents of the `wa_contacts` table: the contacts in the list that are not in the database are those that have been deleted. Note that this procedure works only if the log file reporting the addition of a contact of interest is still available when the analysis is performed.

Unfortunately, by proceeding as above, it is not possible to determine when a given contact has been deleted, since

```
2013-09-27 16:25:09.487 I: [90] xmpp/reader/read/blocklist/add
                                39320XXXXXXX@s.whatsapp.net
```

(a)

```
2013-09-27 16:38:42.313 I: [87] xmpp/writer/write/blocklist
2013-09-27 16:38:42.575 I: [90] general_request_success/3
```

(b)

Fig. 2. Events in the log file corresponding to (a) the blocking, (b) the unblocking of user *39320xxxxxxx*.

Please cite this article in press as: Anglano C, Forensic analysis of WhatsApp Messenger on Android smartphones, Digital Investigation (2014), http://dx.doi.org/10.1016/j.diin.2014.04.003

deletions give rise to log events that do not reference the WhatsApp ID of the contact being deleted.

Analysis of exchanged messages

WhatsApp Messenger stores all the messages that have been sent or received into the chat database `msgstore.db` (located in the directory listed in Table 1, row 2), whose analysis makes it possible to reconstruct the chronology of exchanged messages, namely to determine when a message has been exchanged, the data it carried, the set of users involved in the conversation, and whether and when it has actually been received by its recipients.

In the following we discuss each one of the above steps separately: we start with the description of the structure of the chat database (Sec. The structure of the chat database), and then we explain how to (a) reconstruct the chat history (Sec. Reconstruction of the chat history), (b) determine and extract the content of a message (Sec. Extracting the contents of a message), (c) determine the status of a message (Sec. Determining the state of the message), (d) determine the set of users among which each message has been exchanged (Sec. Determining the partners of a message) and, finally, (e) deal with deleted messages (Sec. Dealing with deleted messages).

The structure of the chat database

The `msgstore.db` database contains the following three tables:

- `messages`, that contains a record for each message that has been sent or received by the user. To ease understanding, we classify the fields of these records in two distinct categories: those storing attributes of the message (listed in Table 3), and those storing the contents of the message and the corresponding metadata (listed in Table 4);
- `chat_list`, that contains a record for each conversation held by the user (a conversation consists into the set of messages exchanged with a particular contact), whose fields are described in Table 5;
- `sqlite_sequence`, that stores housekeeping data used internally by WhatsApp Messenger, whose structure is not reported here since it does not have any evidentiary value.

As reported in (Thakur, 2013), WhatsApp Messenger usually generates various backup copies of the `msgstore.db` database, that are stored in the directory listed in Table 1 row no. 3. These backups are full copies of the `msgstore.db` database, and are not kept synchronized with it. Therefore, they are particularly important from an

Table 3
Structure of the *messages* table: fields storing message attributes.

Field name	Meaning
_id	record sequence number
key_remote_jid	WhatsApp ID of the communication partner
key_id	unique message identifier
key_from_me	message direction: ‘0’ = incoming, ‘1’ = outgoing
status	message status: ‘0’ = received, ‘4’ = waiting on the central server, ‘5’ = received by the destination, ‘6’ = control message
timestamp	time of send if key_from_me = ‘1’, record insertion time otherwise (taken from the local device clock, and encoded as a 13-digits millisecond Unix epoch time)
received_timestamp	time of receipt (taken from the local device clock, and encoded as a 13-digits millisecond Unix epoch time) if key_from_me = ‘0’, ‘−1’ otherwise
receipt_server_timestamp	time of receipt of the central server ack (taken from the local device clock, and encoded as a 13-digits millisecond Unix epoch time) if key_from_me = ‘1’, ‘−1’ otherwise
receipt_device_timestamp	time of receipt of the recipient ack (taken from the local device clock, and encoded as a 13-digits millisecond Unix epoch time) if key_from_me = ‘1’, ‘−1’ otherwise
send_timestamp	unused (always set to ‘−1’)
needs_push	‘2’ if broadcast message, ‘0’ otherwise
recipient_count	number of recipients (broadcast message)
remote_resource	ID of the sender (only for group chat messages)

Table 4
Structure of the *messages* table: fields storing information concerning message contents.

Field name	Meaning
media_wa_type	message type: ‘0’ = text, ‘1’ = image, ‘2’ = audio, ‘3’ = video, ‘4’ = contact card, ‘5’ = geo position)
data	message content when media_wa_type = ‘0’
raw_data	thumbnail of the transmitted file when media_wa_type = {‘1’,‘3’}
media_hash	base64-encoded SHA-256 hash of the transmitted file (when media_wa_type = {‘1’,‘2’,‘3’})
media_url	URL of the transmitted file (when media_wa_type = {‘1’,‘2’,‘3’})
media_mime_type	MIME type of the transmitted file (when media_wa_type = {‘1’,‘2’,‘3’})
media_size	size of the transmitted file (when media_wa_type = {‘1’,‘2’,‘3’})
media_name	name of transmitted file (when media_wa_type = {‘1’,‘2’,‘3’})
media_duration	duration in sec. of the transmitted file (when media_wa_type = {‘1’,‘2’,‘3’})
latitude	latitude of the message sender (when media_wa_type = ‘5’)
longitude	longitude of the message sender (when media_wa_type = ‘5’)
thumb_image	housekeeping information (no evidentiary value)

Please cite this article in press as: Anglano C, Forensic analysis of WhatsApp Messenger on Android smartphones, Digital Investigation (2014), http://dx.doi.org/10.1016/j.diin.2014.04.003

Table 5
Structure of the *chat_list* table.

Field name	Meaning
_id	sequence number of the record
key_remote_jid	WhatsApp ID of the communication partner
message_table_id	sequence number of record in the messages table that corresponds to the last message of the conversation

investigative standpoint, since they may store messages that have been deleted from the main chat database. Backups are encrypted with the AES 192 algorithm, but they can be easily decrypted since, as discussed in (Cortjens et al., 2011), the same encryption key (namely, `346a23652a46392b4d73257-c67317e352e3372482177652c`) is used on all devices.

Reconstruction of the chat history

To reconstruct the chronology of the messages exchanged by the user, the records stored in the `messages` table must be extracted and decoded as discussed below.

To elucidate, let us consider Fig. 3, that shows four records corresponding to a conversation between the device owner and the user `39348`*`xxxxxxx`* (actually, only the fields listed in Table 3 are displayed).

By examining these records, we note that (a) all the messages have been exchanged with the same contact `39348`*`xxxxxxx`* (they all store the same WhatsApp ID in the `key_remote_jid` field), (b) the conversation has been started by that contact (`key_from_me` = '*0*' in record no.1) with a textual message whose content was "*Message 1*" (field `data`) on Feb. 13th, 2012 06:59:09 (field `received_timestamp`), and (c) the device owner replied at 07:00:23 of the same day (field `timestamp`) with the message corresponding to record no. 2 (`key_from_me` = '*1*') with content "*Reply 1*" (field `data`). The conversation then continued with another message-reply exchange.

From these records, we also note that each message carries its own unique identifier in the `key_id` field: this value, set by the sender, is obtained by concatenating the timestamp corresponding to the last start time of WhatsApp Messenger (on the sender's device) with a progressive number (indicating the number of messages sent from that moment), and is used also by the recipient to denote that message. Therefore, by using this value, it is possible to correlate the records of the sender's and recipient's databases corresponding to the same message.

Extracting the contents of a message

In addition to plain text messages, WhatsApp allows its users to exchange messages containing data of various types, namely multimedia files (storing images, audio, and video), contact cards, and geo-location information. The type of data transmitted with a message is indicated (as reported in Table 4) by the `media_wa_type` field, while the information concerning message content is spread, for non-textual messages, over several fields (depending on the specific data type). As a matter of fact, while the content of textual messages (`media_wa_type` = '0') is stored in the `data` field, for the other types of contents the situation is more involved, as discussed below.

Multimedia files. When the user sends a multimedia file, several activities take place automatically (i.e., without informing the involved users). First, WhatsApp Messenger copies the file into the folder listed in Table 1, row 8. Then, it uploads the file to the WhatsApp server, that sends back the URL of the corresponding location. Finally, the sender sends to the recipient a message containing this URL and, upon receiving this message, the recipient sends an acknowledgment back to the sender.

When these steps have been completed, the sender stores into his/her `messages` table a record like the one shown in Fig. 4 (where we show only the fields related to message contents that are listed in Table 4). As can be seen from the above figure, the type of the file is indicated (besides the `wa_media_type` field, not shown in the figure) by the `media_mime_type` field ('*image/jpeg*' in the example). Its name is instead stored in the `media_name` field (*IMG-20131021-WA0000.jpg* in the example), its size in bytes by `media_size` (*40267* in the example), and its thumbnail in the `raw_data` field (as a *blob*, i.e. a binary large object). Furthermore, the `media_url` field stores the URL of the location on the central server where the file has been temporarily stored, whose last part (highlighted in Fig. 4 by framing it) corresponds to the name given by the server to that file. Finally, the base64-encoded SHA-256 hash of the transmitted file is stored in the `media_hash` field.

On the recipient side, after message reception, the transmitted thumbnail of the file is displayed by WhatsApp Messenger; the actual file is instead downloaded at a later time only if the recipient explicitly requests it. Upon message reception, the recipient stores in his/her messages table a record like the one shown in Fig. 5. From this figure, we see that most fields are identical to those stored by the sender (in particular, `wa_media_type`, `media_mime_type`, `media_size`, `raw_data`, and `media_hash`). Conversely, the contents of `media_url` is different, except

	key_id	key_remote_jid	key_from_me	timestamp	received_timestamp	data
1	1329115800-1	39348 @s.whatsapp.net	0	1329116347000	1329116349643	Message 1
2	1329116349-1	39348 @s.whatsapp.net	1	1329116423505	1329116423532	Reply 1
3	1329115800-2	39348 @s.whatsapp.net	0	1329116791000	1329116793357	Message 2
4	1329116349-2	39348 @s.whatsapp.net	1	1329116941607	1329116941626	Reply 2

Fig. 3. Reconstruction of the chat history. Phone numbers have been grayed out to ensure the privacy of the owner.

Please cite this article in press as: Anglano C, Forensic analysis of WhatsApp Messenger on Android smartphones, Digital Investigation (2014), http://dx.doi.org/10.1016/j.diin.2014.04.003

Fig. 4. Multimedia file exchange: sender side.

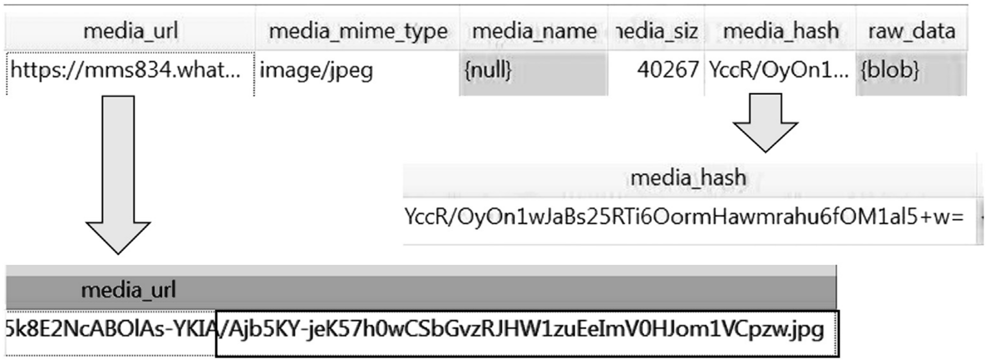

Fig. 5. Multimedia file exchange: recipient side.

for the name given to the file by the server (highlighted in Fig. 5 by framing it).

Unlike the sender, however, the `media_name` field is empty, so the local name given by WhatsApp Messenger to that file is unknown. The file can be however identified by comparing the SHA-256 hash stored into the corresponding record with that of all the files that have been received that are stored in the folder reported in Table 1, row 7.

Finally, we note that the file sent by the sender and the one received by the recipient can be correlated by comparing both the file name given by the WhatsApp server and the SHA-256 hash to these files (that are stored, as discussed above, in the `media_url` and `media_hash` fields of the corresponding records).

Contact cards. Messages carrying contacts cards (extracted from the phonebook of the sender) correspond – both on the sender and on the recipient side – to `messages` record that store the transmitted information (in `VCARD` format) into the `data` field, and the name given by the sender to that contact in the `media_name` field. An example of such a record is shown in Fig. 6.

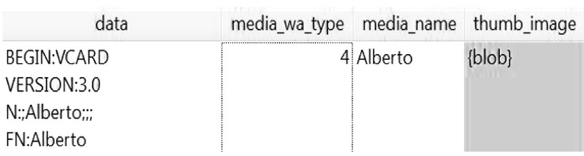

Fig. 6. Fields containing contact card information.

Geolocation coordinates. WhatsApp Messenger enables users to send the geographic coordinates of their current location, that are obtained from the Android Location Services running on the device. Messages carrying geographic coordinates correspond –both on the sender and on the recipient side – to `messages` records that store the latitude and the longitude values into the `latitude`, `longitude` fields, and a JPEG thumbnail of the Google Map displaying the above coordinates in the `raw_data` field. An example of such a record is shown in Fig. 7.

Determining the state of the message

In WhatsApp, messages are not exchanged directly among communicating users, but they are first sent to the central server, that forwards them to the respective recipients if they are on-line, and stores them locally until they can be delivered otherwise. This implies that the presence of a record in the `messages` table does not necessarily mean that an outgoing message has been

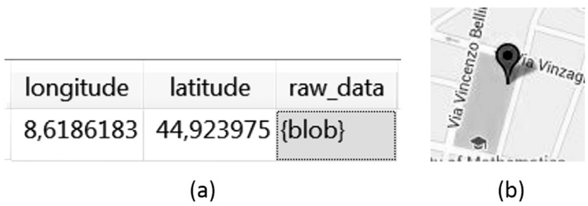

Fig. 7. Geo-location message:(a) data stored in the database, (b) Google map extracted from the `raw_data` field.

key_id	status	timestamp	received_timestamp	receipt_server_timestamp	receipt_device_timestamp
1381932918-1	0	1381932937884	1381932937888	-1	-1

(a) both sender and recipient offline

key_id	status	timestamp	received_timestamp	receipt_server_timestamp	receipt_device_timestamp
1381932918-1	4	1381932937884	1381932937888	1381933025551	-1

(b) sender becomes on line, recipient still offline

key_id	status	timestamp	received_timestamp	receipt_server_timestamp	receipt_device_timestamp
1381932918-1	5	1381932937884	1381932937888	1381933025551	1381933319135

(c) recipient becomes online

Fig. 8. Sender side: record updates for a message while in transit.

actually delivered to its recipients. As a matter of fact, after the user has pushed the "send" button of WhatsApp Messenger, the message can be in one of the following three states: (a) waiting on the local device to be transmitted to the central server, or (b) stored on the central server but waiting to be transmitted to its recipient(s), or (c) delivered to its recipient(s).

The ability to distinguish the various states of a message may be crucial in an investigation where it must be ascertained whether a message has been actually delivered or not to its destinations, and when such a delivery has taken place.

The current state of a message, as well as the times of its state changes, can be understood by correlating the values contained in several fields of the corresponding record of the sender database,[3] namely `status`, `timestamp`, `received_timestamp`, `receipt_server_timestamp`, and `receipt_device_timestamp`.

To explain, let us consider a scenario in which a user sends a message when both him/her and the recipient are off-line (Fig. 8(a)), then the sender gets reconnected to the network while the recipient is still offline (Fig. 8(b)), and then, finally, also the recipient gets connected (Fig. 8(c)).

When the message is sent, a record is stored in the `messages` table of the sender, even if the central server is unreachable. In this case, as shown in Fig. 8(a), in this record we have that `status` = '0', `timestamp` = '*x*', and `received_timestamp` = '*y*', where '*x*' and '*y*' correspond to when the user has sent the message and when the record has been added to the chat database, respectively.

Thus, a record such that `key_from_me` = *'1' and* `status` = *'0' corresponds to a message that has not been delivered to the central server yet.*

Later, when the sender returns on-line, the message is forwarded to the central server that replies with an ack. When this ack is received, the sender updates the corresponding record as shown in Fig. 8(b) by setting `status` = '4', and the value of `receipt_server_timestamp` to the reception time of the ack.

Thus, a record such that `key_from_me` = *'1' and* `status` = *'4' corresponds to a message that has been delivered the central server, but not yet to its destination(s).*

Finally, when the recipient returns on line, it receives the message from the central server, and sends an ack to the sender. Upon receiving this ack, the sender updates again the record corresponding to that message (as shown in Fig. 8(c)) by setting `status` = '5', and the value of `receipt_device_timestamp` to the reception time of the ack.

Thus, a record such that `key_from_me` = *'1' and* `status` = *'5' corresponds to a message that has been delivered to its destination.*

From the above discussion, it follows that the times of the state changes of a message can be tracked by means of the values stored in the various `timestamp` fields of the corresponding record. For instance, in the case in Fig. 8(c), we can deduce that the message has been generated on Oct. 16th, 2013 14:15:37.884 (`timestamp` field), has been waited to be transmitted to the central server until 14:17:05.551 of the same day (`receipt_server_timestamp` field), and has been finally delivered to its recipient at 14:21:59.135 (`receipt_device_timestamp` field).

Determining the partners of a message

In addition to user-to-user communication, WhatsApp provides its users with two forms of collective communications, namely:

- *broadcast* (i.e, one-to-many) communication, whereby a user (the *source user*) sends the same message to a group of other users (the *destination users*) that are not aware of each other and whose possible replies are sent to the source user only;
- *group chats*, providing a many-to-many communication service, whereby each message sent by any user belonging to a chat is received by all the users belonging to that chat.

While the WhatsApp ID of the communication partner in a user-to-user communication is easily retrieved from

[3] For a recipient, a message can be only in the received state, corresponding to status = '0'.

Please cite this article in press as: Anglano C, Forensic analysis of WhatsApp Messenger on Android smartphones, Digital Investigation (2014), http://dx.doi.org/10.1016/j.diin.2014.04.003

	key_id	key_remote_jid	remote_resource	recipient_count	needs_push
1	1382694005-1	39320 @s.whatsapp.net	39320 @s.w...	3	2
2	1382694005-1	39335 @s.whatsapp.net	39320 @s.w...	3	2
3	1382694005-1	39333 @s.whatsapp.net	39320 @s.w...	3	2
4	1382694005-1	broadcast	39320 @s.w...	3	2

(a)

key_id	key_remote_jid	remote_resource	recipient_count	needs_push
%~1382694005-1	39320 @s.whatsapp.net		{null}	0

(b)

Fig. 9. Records generated for a broadcast message sent to three recipients on: (a) the sender, (b) one of the recipients. Only the fields that contribute to the identification of the partners are displayed.

the `key_remote_jid` field, to determine the set of users involved into a broadcast or a group chat message various fields have to be correlated, as discussed below.

Broadcast messages. When a user sends a broadcast message, a distinct record is created in his/her `messages` table for each one of the recipients, plus one for itself, as reported in Fig. 9(a), that shows the records generated by a broadcast message sent to users *39320xxxxxxx*, *39335xxxxxxx*, and *39333xxxxxxx*.

As shown in this figure, all the records corresponding to the same broadcast message have the same message identifier (stored in the `key_id` field), so they can be easily identified. Each one of these records stores in the `key_remote_jid` field the WhatsApp ID of the recipient (the sender uses the keyword *broadcast* to denote itself as a recipient), while the `remote_resource` and the `recipient_count` fields store the WhatsApp IDs of the set of destinations and how many they are, respectively (field `needs_push` instead always stores the value '*2*').

The situation on each one of the destinations is instead different (Fig. 9(b)), since each one of them stores, in his/her `messages` table, only a single record that is generated when it receives the broadcast message. This record can be distinguished from those corresponding to non-broadcast messages by looking at the value stored in its `key_id` field, that consists in the concatenation of the *%~* characters with the message identifier set by the sender.

Group chat communication. When a message is sent within a group chat, a record is generated in the `messages` table of all the members of that group (including the sender). Each one of these records stores, in the `key_remote_jid` field, the identifier of the group (the `group_id`), a string formatted as `{creator's phone number}-{creation time}@g.us` (where *creation time* is encoded as a Unix epoch time).

To illustrate, consider a group chat consisting of three members, namely `3933xxxxxxx`, `3936xxxxxxx`, and `3932xxxxxx` (in the following denoted as *A*, *B*, and *C*, respectively, for brevity), where each user sends in turn to the group a message with textual content '*Message from X*' (where 'X' is the name of the user).

Let us focus on the records stored in the `messages` table of user *A* at the end of this exchange, that are shown in Fig. 10 (the situation for the other users is identical).

As can be seen from this figure, all these records store the same `group_id` *3933xxxxxxx-1363078943@g.us* in the `key_remote_id` field. From this value, we can determine the creator of the group (user *A*) and the date and hour of group creation (March 12, 2013 at 09:02:23). Furthermore, the WhatsApp ID of the message originator is stored into the `remote_resource` field, while the time of message receipt is stored into the `timestamp` field. Note that *A* stores also the record corresponding to the message that (s)he has sent to the group (see record no. 1 in Fig. 10). Records like this one can be easily identified by looking at the contents of their `status` and `remote_resource` fields, that store the values '*4*' and '*null*', respectively.

Note also that the set of recipients, i.e. of the set of group members at the time of the sending, is not stored anywhere on the record. However, it can be indirectly determined by examining the records corresponding to the *control messages* that are automatically exchanged by the various group members each time a user joins or leaves the group.

	key_remote_jid	remote_resource	key_from_me	status	timestamp	data
1	3933 -1363078943@g.us	{null}	1	4	1363079028764	Message from A
2	3933 -1363078943@g.us	3936 @s.whatsapp.net	0	0	1363079064000	Message from B
3	3933 -1363078943@g.us	3932 @s.whatsapp.net	0	0	1363079078000	Message from C

Fig. 10. Records corresponding to three messages exchanged within a group chat.

	key_remote_jid	remote_resource	key_from_me	status	media_size	data	timestamp	received_timestamp
1	39320[redacted]1384187045@g.us	39320[redacted]@s.whatsapp.net	1	6	1	wa test group	1384187045000	1384187045496
2	39320[redacted]1384187045@g.us	39335[redacted]@s.whatsapp.net	1	6	4	{null}	1384187045970	1384187046015
3	39320[redacted]1384187045@g.us	39333[redacted]@s.whatsapp.net	1	6	4	{null}	1384252848773	1384252848834
4	39320[redacted]1384187045@g.us	39333[redacted]@s.whatsapp.net	1	6	5	{null}	1384467096987	1384467097066
5	39320[redacted]1384187045@g.us	39335[redacted]@s.whatsapp.net	1	6	5	{null}	1384508994642	1384508994761

Fig. 11. Group management records stored in the msgstore database of user *D*. For other users we have the same situation, with the exception of record no. 1.

These messages, also stored in the `messages` table, always contain the value '6' in the status field, and encode in the `media_size` field the specific operation corresponding to the message (in particular, the values '*1*', '*4*', and '*5*' indicate group creation, join, and leave, respectively).

To illustrate, let us consider a scenario in which user *39320xxxxxxx* (*D*, for brevity) creates a group on Nov. 11, 2013 at 16:24:05, and immediately adds user *39335xxxxxxx* (*E*, for brevity) to the group. Then, *D* adds user *39333xxxxxxx* (*F*, for brevity) on Nov. 12, 2013 at 10:40:48.

The records generated by these operations in the chat database of user *D* are shown in Fig. 11 (the same situation occurs on all the other group members).

Group creation corresponds to record no. 1, as can be seen from status = '6' and `media_size` = '*1*'. The time of group creation can be ascertained (besides from the `group_id`) from the value stored in the `timestamp` field, while the field `data` stores the name given to the group (*wa test group*).

The addition of user *E* corresponds instead to record no. 2: the specific operation (join) and the identity of the user joining the group (*E*) can be deduced from fields `media_size` and `remote_resource` field, while the time of occurrence is stored in the `timestamp` field. A similar situation occurs with the addition of user *F*, whose control message corresponds to record no. 3.

Now, suppose that at a later time, namely on Nov. 14, 2013 at 22:11:36, user *F* leaves the group. This operation corresponds to record no. 4 in Fig. 11 (media_size = '5' indicates a group leave), that reports the identity of the user leaving the group and the time of leave in the remote_resource and the timestamp field, respectively. Finally, when user *E* leaves the group on Nov. 15, 2013 at 09:49:54, record no. 5 is added to the `messages` table.

By using the information discussed above, the composition of the group over time can be reconstructed by chronologically sorting the various control messages corresponding to join (`status` = '6' and `media_size` = '4') and leave (`status` = '6' and `media_size` = '5') of a given group (identified by the contents of the `key_remote_jid` field), as shown in Fig. 12.

From this information, it can be inferred whether a user belonged or not to a group when a specific message was sent to that group.

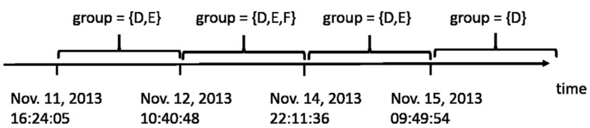

Fig. 12. Timeline of variations in group membership.

Dealing with deleted messages

In WhatsApp Messenger, the user may delete the records stored in the `msgstore.db` database in two different ways, namely:

- deletion of an individual message: in this case, the corresponding record is deleted from the `messages` table;
- deletion of all the records belonging to a one-to-one, broadcast, or group chat conversation: in this case, all the records corresponding to the messages exchanged in that conversation are deleted from the `messages` table, as well as the record of the `chat_list` table corresponding to that conversation.

As discussed before, it is sometimes possible to recover deleted SQLite records, and in these cases the analysis techniques discussed in the previous sections can be applied.

However, when such a recovery is not feasible, it may be still possible to determine many of the information regarding a deleted message by analyzing the log files generated by WhatsApp Messenger. In particular, as discussed below, it is possible to determine which messages have been deleted and when, when a deleted message has been sent or received and its state, as well as the users involved in the conversation. The same holds true for group control messages, so the analysis of log files also makes it possible to track the evolution of each group over time. In other words, only the contents of a deleted message cannot be recovered anymore.

Finding deleted messages and their deletion times. When a message is deleted, WhatsApp Messenger records into the log file an event like the one shown in Fig. 13, that indicates both the type of operation (`msgstore/delete`) and the identifier of the deleted message (*1363253484-1*), as well as the time of deletion (March 14, 2013 at 10:49:22).

Determining when a deleted message has been exchanged, and its state. Each time a user-to-user, broadcast, or group chat message is sent/received, WhatsApp Messenger logs the time of the send/receive operation, the identifiers of the involved users, and the identifier of the message. Therefore, by searching into the log file the events corresponding to exchanges of deleted messages, it is possible to ascertain when those messages have been sent or received.

For instance, the event reported in Fig. 14 line 1 corresponds to the sending of the deleted message identified by

```
[...]
2013-03-14 10:49:22.737 I: msgstore/delete 1363253484-1
```

Fig. 13. Events logged when a message is deleted.

```
1  2013-03-14 09:37:44.621 I: msgstore/add/send 39366XXXXXXX@s.whatsapp.net
2                              1363253484-1 0
3  2013-03-14 09:37:44.648 I: xmpp/writer/write/message 39366XXXXXXX@s.whatsapp.net
4                              1363253484-1 | time: 1
5  2013-03-14 09:37:44.887 I: xmpp/reader/read/message/receipt/server
6                              39366XXXXXXX@s.whatsapp.net 1363253484-1
7  2013-03-14 09:37:45.190 I: xmpp/reader/read/message/receipt/device
8                              39366XXXXXXX@s.whatsapp.net 1363253484-1
```

Fig. 14. Events logged when a message is sent.

1363253484-1 to user *39366xxxxxxx* on March 14, 2013 at 09:37:44.

Finally, WhatsApp Messenger logs also the events corresponding to reception of the acknowledgment messages sent back by the central server (line no. 5) and by the recipient (line no. 7), from which it is possible to determine the state of a message, as well as the times of its state changes.

Temporal evolution of group chat composition. The evolution of the composition of a group chat can be tracked over time by examining the events, logged by WhatsApp Messenger, corresponding to the exchange of the various control messages discussed in Sec. Determining the partners of a message.

The events corresponding to the creation of a group and to the addition of two users are shown in Fig. 15 (that reports an excerpt of the log of the group creator).

The creation of the group gives rise to the events reported in lines no. 1 and 4, from which we can obtain the group name and creation time. The request to add to the group users *39366xxxxxxx* and *39320xxxxxxx* corresponds to line 7, while the addition of the former and the latter user corresponds to lines 10–14 and 18–22. Finally, the leave of a user corresponds to the event logged on line no. 27.

Analysis of settings and preferences

WhatsApp Messenger stores various information of potential evidentiary value in several files, located in the directories listed in Table 1 row no. 9.

In particular, the file `me` stores (as ASCII text) the phone number registered with WhatsApp (i.e., the number used to create the corresponding WhatsApp ID). The relevance of this information derives from the fact that the SIM card currently used with the smartphone may not be the one used to register the user with WhatsApp: it is indeed possible to replace the latter SIM card with a new one, and to use the WhatsApp ID corresponding to the phone number of the old SIM card. Thus, a user *A* may impersonate a different user *B*, as long as *A* has used *B*'s SIM card during registration, or (s)he is using *B*'s smartphone with a different SIM card. By comparing the phone number of the SIM inserted into a smartphone with the phone number stored in the `me` file, it is possible to determine whether this is the case or not.

Furthermore, the file `me.jpg` stores the currently-used avatar picture of the user. Given that the avatar pictures of all contacts are downloaded locally by WhatsApp

```
1  2013-03-12 09:02:24.523 I: newgroup/go create group:
2                              39333XXXXXXX-prova gruppo@temp
3  [...]
4  2013-03-12 09:02:25.146 W: groupmgr/onGroupCreated/gid:
5                              39333XXXXXXX-1363078943@g.us/prova gruppo/1363078943000
6  [...]
7  2013-03-12 09:02:25.204 E: xmpp/writer/write/req-add-participants 39333XXXXXXX-1363078943@g.us |
8                              [39366XXXXXXX@s.whatsapp.net, 39320XXXXXXX@s.whatsapp.net]
9  [...]
10 2013-03-12 09:02:25.453 I: groupmgr/onGroupAddUser/jid:
11                             39366XXXXXXX@s.whatsapp.net/gid:39333XXXXXXX-1363078943@g.us
12 2013-03-12 09:02:25.454 I: groupmgr/addParticipantToList:[39333XXXXXXX-1363078943@g.us |
13                             39366XXXXXXX@s.whatsapp.net] old pas:
14 2013-03-12 09:02:25.454 I: groupmgr/addParticipantToList/update:39366XXXXXXX@s.whatsapp.net
15 2013-03-12 09:02:25.455 I: groupmgr/handle_add_groupchat_msg
16 2013-03-12 09:02:25.456 I: msgstore/add/send 39333XXXXXXX-1363078943@g.us 1363079263-5 0
17 [...]
18 2013-03-12 09:02:25.459 I: groupmgr/onGroupAddUser/jid:
19                             39320XXXXXXX@s.whatsapp.net/gid:39333XXXXXXX-1363078943@g.us
20 2013-03-12 09:02:25.459 I: groupmgr/addParticipantToList:[39333XXXXXXX-1363078943@g.us |
21                             39320XXXXXXX@s.whatsapp.net] old pas:39366XXXXXXX@s.whatsapp.net
22 2013-03-12 09:02:25.459 I: groupmgr/addParticipantToList/update:39366XXXXXXX@s.whatsapp.net,
23                             39320XXXXXXX@s.whatsapp.net
24 2013-03-12 09:02:25.461 I: groupmgr/handle_add_groupchat_msg
25 2013-03-12 09:02:25.461 I: msgstore/add/send 39333XXXXXXX-1363078943@g.us 1363079263-6 0
26 [...]
27 2013-03-12 09:11:03.875 I: groupmgr/onGroupRemoveUser/jid:
28                             39320XXXXXXX@s.whatsapp.net/gid:39333XXXXXXX-1363078943@g.us/
29                             removed_by:null
```

Fig. 15. Events corresponding to group operations.

Please cite this article in press as: Anglano C, Forensic analysis of WhatsApp Messenger on Android smartphones, Digital Investigation (2014), http://dx.doi.org/10.1016/j.diin.2014.04.003

Messenger (as discussed in Sec. Retrieving contact information), the me.jpg file can be used to understand that the user of the device under examination has been in contact with another user even if the latter one has deleted from its contacts database the record corresponding to the former one. As a matter of fact, the deletion of a record from the contacts database does not cause the deletion of his/her downloaded avatar picture.

Conclusions

In this paper we have discussed the forensic analysis of the artifacts left by WhatsApp Messenger on Android smartphones, and we have shown how these artifacts can provide many information of evidentiary value. In particular, we have shown how to interpret the data stored into the contacts and chat databases in order to reconstruct the list of contacts and the chronology of the messages that have been exchanged by users.

More importantly, we have also shown the importance of correlating among them the artifacts generated by WhatsApp Messenger in order to gather information that cannot be inferred by examining them in isolation. As a matter of fact, while the analysis of the contacts database makes it possible to reconstruct the list of contacts, the correlation with the events stored into the log files maintained by WhatsApp Messenger allows the investigator to infer also when a specific contact has been added, or to recover deleted contacts and their time of deletion. Similarly, the correlation of the contents of the chat database with the information stored into the log files allows the investigator to determine which messages have been deleted, when these messages have been exchanged, and the users that exchanged them.

The results reported in this paper have a two-fold value. On the one hand, they provide analysts with the full picture concerning the decoding, interpretation, and correlation of WhatsApp Messenger artifacts on Android devices. On the other hand, they represent a benchmark against which the ability of extraction tools for smartphone to retrieve all the WhatsApp Messenger artifacts can be assessed.

It is however worth to point out that the results discussed in this paper apply to Android only: as a matter of fact, there is evidence (Tso et al., 2012) showing that on different platforms (e.g., iOS) WhatsApp Messenger produces artifacts that differ either in the information they store, or in their format, or in both. We leave the analysis of WhatsApp Messenger for other smartphone platforms as future work.

References

AccessData Corporation. FTK imager. Available at, http://www.accessdata.com/support/product-downloads; 2013.

Barghuthi NA, Said H. Social networks IM forensics: encryption analysis. J Commun 2013;8(11).

Bellovin SM, Blaze M, Clark S, Landau S. Lawful hacking: using existing vulnerabilities for wiretapping on the internet. In: Proceedings of Privacy Legal Scholars Conference. Available at, http://dx.doi.org/10.2139/ssrn.2312107; 2013.

Cellebrite LTD.. Cellebrite Android forensics. Available at, http://www.cellebrite.com/mobile-forensics/capabilities/android-forensics; 2013.

Cortjens D, Spruyt A, Wieringa W. WhatsApp database encryption project report. Technical Report. Available at, https://www.os3.nl/_media/2011-2012/students/ssn_project_report.pdf; 2011.

Ho D. Notepad++ Home. Available at, http://notepad-plus-plus.org; 2013.

Hoog A. Android forensics: Investigation, analysis and mobile security for Google Android. Elsevier – Syngress; 2011.

Husain M, Sridhar R. iForensics: forensic analysis of instant messaging on smart phones. In: Goel S, editor. Digital forensics and cyber crime, Vol. 31. Heidelberg: Springer Berlin; 2010. Lecture notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering.

Jeon S, Bang J, Byun K, Lee S. A recovery method of deleted records for SQLite database. Personal Ubiquotous Comput 2012;16.

Mahajan A, Dahiya M, Sanghvi H. Forensic analysis of instant messenger applications on Android devices. Int J Comput Appl 2013;68(8).

Micro Systemation, XRY. Available at, http://www.msab.com/xry/xry-current-version; 2013.

Olivarez-Giles N. Whatsapp adds voice messaging as it hits 300 million monthly active users. The Verge; 2013. Available at, http://www.theverge.com/2013/8/6/4595496/whatsapp-300-million-active-users-voice-messaging-update.

Oracle Corp.. Oracle VM Virtual Box http://www.virtualbox.org; 2013.

Oxygen Forensics, Inc.. Oxygen Forensics. Available at, http://www.oxygen-forensic.com/en/features/analyst; 2013a.

Oxygen Forensics, Inc.. SQLite Viewer. Available at, http://www.oxygen-forensic.com/en/features/analyst/data-viewers/sqlite-viewer; 2013b.

Pooters I, Arends P, Moorrees S. Extracting SQLite records: Carving, parsing and matching. Technical report; Digital Forensics Research Workshop Challenge. Available at, http://sandbox.dfrws.org/2011/fox-it/DFRWS2011_results/Report/Sqlite_carving_extractAndroidData.pdf; 2011.

Sangiacomo F, Weidner M. WhatsApp Xtract (v. 2.1). Available at, https://code.google.com/p/hotoloti/downloads/list; 2012.

SQLite Consortium. SQLite Home Page. Available at, http://www.sqlite.org; 2013.

Thakur N. Forensic analysis of WhatsApp on Android smartphones. Master's thesis. University of New Orleans; 2013. Paper 1706.

The United Nations Office on Drugs and Crime. Comprehensive study on Cybercrime. Technical Report. United Nations; 2013. Available at, http://www.unodc.org/documents/organized-crime/UNODC_CCPCJ_EG.4_2013/CYBERCRIME_STUDY/_210213.pdf.

Tso YC, Wang SJ, Huang CT, Wang WJ. iPhone social networking for evidence investigations using iTunes forensics. In: Proceedings of the 6th International Conference on Ubiquitous information management and Communication. New York, NY, USA: ACM; ICUIMC'12; 2012.

Vanek P, Les K. Sqlite databases made easy. Available at, http://sqliteman.com; 2013.

Vidas T, Zhang C, Christin N. Towards a general Collection methodology for Android devices. Digit Investig 2011;8.

WhatsApp Inc.. WhatsApp. Available at, http://www.whatsapp.com; 2013.

Winkler R. WhatsApp hits 400 million users, wants to stay independent. Wall Str J – Digits. Available at, http://blogs.wsj.com/digits/2013/12/19/whatsapp-hits-400-million-users-wants-to-stay-independent; 2013.

YouWave Corp.. Youwave home page. Available at, http://youwave.com; 2013.

Please cite this article in press as: Anglano C, Forensic analysis of WhatsApp Messenger on Android smartphones, Digital Investigation (2014), http://dx.doi.org/10.1016/j.diin.2014.04.003