Forensic analysis of Kik messenger on iOS devices

Ovens, Kenneth M.; Morison, Gordon

Published in:
Digital Investigation

DOI:
10.1016/j.diin.2016.04.001

Publication date:
2016

Document Version
Peer reviewed version

Citation for published version (Harvard):
Ovens, KM & Morison, G 2016, 'Forensic analysis of Kik messenger on iOS devices', *Digital Investigation*, vol. 17, pp. 40–52. https://doi.org/10.1016/j.diin.2016.04.001

Forensic analysis of Kik messenger on iOS devices

Kenneth M. Ovens*, Gordon Morison

School of Engineering & Built Environment, Glasgow Caledonian University, Cowcaddens Road, Glasgow, G4 0BA, Scotland.

Abstract

Instant messaging applications continue to grow in popularity as a means of communicating and sharing multimedia files. The information contained within these applications can prove invaluable to law enforcement in the investigation of crimes.

Kik messenger is a recently introduced instant messaging application that has become very popular in a short period of time, especially among young users. The novelty of Kik means that there has been little forensic examination conducted on this application.

This study addresses this issue by investigating Kik messenger on Apple iOS devices. The goal was to locate and document artefacts created or modified by Kik messenger on devices installed with the latest version of iOS, as well as in iTunes backup files. Once achieved, the secondary goal was to analyse the artefacts to decode and interpret their meaning and by doing so, be able to answer the typical questions faced by forensic investigators.

A detailed description of artefacts created or modifed by Kik messenger is provided. Results from experiments showed that deleted images are not only recoverable from the device, but can also be located and downloaded from Kik servers. A process to link data from multiple database tables producing accurate chat histories is explained. These outcomes can be used by law enforcement to investigate crimes and by software developers to create tools to recover evidence.

Keywords: Kik, Instant messaging, iOS, Mobile device forensics, Apple

*Corresponding author:
Email address: kenneth.ovens@gcu.ac.uk (Kenneth M. Ovens)

1. Introduction

Instant messaging is not new; in fact, it has been claimed to be older than the Internet itself (Van Vleck, 2012). The popularity of messaging applications grew in the 1990's when graphical user interfaces replaced text-based interfaces. At that time, the popular applications included AOL Instant Messenger, ICQ and Yahoo! Messenger.

What is relatively novel is the popularity they have gained on the mobile platform. Just as smartphones and tablets overtake laptops and personal computers as the most popular method of accessing the Internet, instant messaging applications are significantly gaining ground on traditional phone calls and text messaging as the favoured means of communication, especially for the younger generation (Ofcom, 2015).

There are now billions of instant messaging user accounts; currently the most popular applications include WhatsApp, Facebook Messenger, Skype, and Viber. A more recent addition to instant messaging, and one that is especially popular among younger users, is the application, Kik. Launched in 2010, Kik's user base has currently grown to over 200 million, including 40% of American youth, according to the developer's website (Kik, 2015b).

As Kik has grown in popularity, crimes that have in some way involved the application, have also increased, particularly crimes that involve bullying and child abuse (Alvarez, 2013; Federal Bureau of Investigation, 2015; Zauzmer, 2014). These types of crimes are not unique to Kik, but weak user identification, no age verification, as well as user's perceived anonymity, may be combining to create user behaviours that are of concern to law enforcement (Godfrey, 2013; Larson, 2015).

During registration of a new account, the user is prompted to submit a first and last name, a unique username, email address, password, and a date of birth. However, there is no requirement to link a mobile phone number and failure to verify the email address does not prohibit the user sending messages. In comparison to registering a new account with Facebook, where it is a requirement to use your real name, if email addresses are not verified, the accounts can not continue to be used. New account registrations for WhatsApp and Viber require phone numbers to be linked and verified. While it is not too difficult for a determined person to bypass these verification steps, there is little effort required to bypass Kik's verification procedures and age restrictions. Kik states that users are required to be at least thirteen years old, as this also is not verified, it is very easy for younger users to enter a fake date of birth and begin communicating immediately (Kik, 2015c).

What will be of further concern to law enforcement, is that Kik do not store and cannot retrieve any sent or received messages (Kik, 2015a). It is therefore crucial that forensic examiners are able to obtain as much information as possible from recovered mobile devices to aid investigations. While there has been a growing body of research concerning the more established instant messaging applications, to date, there is a distinct lack of detailed forensic investigation focused on Kik messenger.

The situation prompts this study into the identification, recovery, and analysis of artefacts relating to the usage of Kik messenger. This study provides the first detailed forensic analysis of Kik on Apple iOS devices. Other platforms on which Kik can be installed (Android, Windows, Amazon) are outside the scope of this study and are left for future work.

Preconditions to accessing these artefacts are that the iOS device is not password locked and the investigator has access to unencrypted backup files.

The following proposed questions, common to forensic examinations, are the focus of this study:

1. Who has the user been communicating with and when?
2. What was the content of the communications?
3. What attachments were exchanged and where can they be found?

The study was conducted using tools that are freely available to practitioners. The results were used to develop open-source software that can be used to extract and present Kik artefacts, and can likewise be used by other software developers to create more forensic tools that can accurately retrieve relevant data. It also contributes to the documentation and analysis of artefacts created by Kik messenger, benefiting law enforcement in their investigations.

The rest of this paper is structured as follows: Section 2 presents an overview of research conducted into instant messaging applications and discusses methods used to acquire data from iOS devices. Section 3 describes the experiments undertaken to address the typical questions that would arise in a forensic investigation. Section 4 reports the results and analysis of the experiments. Finally, Section 5 draws conclusions from the study and proposes avenues for future research.

2. Related Work

A brief summary of the research methods, limitations, and conclusions for each study has been provided.

Early instant messaging research on mobile devices focused on the popular applications of the time. Husain and Sridhar (2010) examined artefacts from three applications: AIM, Yahoo! and Google Talk. With a limited data set (two messages for each application), the authors located artefacts that could have been of evidentiary value. This was achieved by searching the backup files of an iPhone 3G which had a firmware version (later named iOS) of 2.2.1. The backup files were produced by Apple's mobile device management application, iTunes.

A more comprehensive examination of the iTunes backup data was performed by Bader and Baggili (2010), to establish what data of forensic value could be recovered. The researchers manually searched through the backup files of an iPhone 3GS installed with firmware version 3.1.2, and located various types of data using command-line tools such as *'grep'* and *'find'*. The iTunes backup files were then matched against the original files located on the iPhone. This research is useful and can be applied to studies of any application that is backed up by iTunes.

Al Mutawa et al. (2012) researched social networking applications that also offer instant messaging features, namely Facebook, MySpace, and Twitter. The devices examined were iPhone 4 (iOS 4.3.3), Android, and BlackBerry mobile phones. The methods employed for the study involved installing the social networking applications on the devices, performing common user activities, then obtaining a logical image of each device before conducting a manual analysis. For the iOS device, the researchers used the iTunes application to obtain a backup of the user files. From this, they were able to extract artefacts relating to the social networking applications.

Tso et al. (2012) also focused on examining social networking applications and chose the most popular applications at that time, Facebook Chat, Viber, Skype, WhatsApp, and Windows Live Messenger. One of the reasons stated as justification for the study was that the applications provide instant and convenient information transmission used by criminals. The researchers examined an iPhone 4 with iOS 4.3.5 installed. Again, the iTunes backup application was leveraged to acquire the relevant artefacts.

Sgaras et al. (2015) also highlighted the growing concern of instant messaging applications being used by criminals to communicate with victims or to evade detection. The researchers suggested that published studies focused mainly on Android devices, whereas iOS devices had not been extensively examined. Commercial tools, namely Cellebrites UFED (Universal Forensic Extraction Device), were used to extract and classify data from messaging applications WhatsApp, Viber, Skype, and Tango, on an iPhone (iOS 6.1.3) and an Android device.

Comparing various data extraction methods available at the time of the study, Hay et al. (2011) concluded that in order to perform a comprehensive examination of an iOS device, it would need to be jailbroken and manually analysed. This method has raised concerns regarding changes made to the device during the jailbreaking process (Husain et al., 2011; Piccinelli and Gubian, 2011). However, it could be used initially, to gain an understanding of Kik messenger's mechanics by observing the creation and modification of data as the application is used.

As with most research in digital forensics, these studies have focused mainly on the forensic acquisition of data from devices, with less emphasis on the analysis and interpretation of that data. Studies that have investigated how recovered data could be interpreted and applied to criminal investigations include Levinson et al. (2011). This study used a mock scenario where Facebook Chat artefacts found on an iOS device provided key evidence in a police investigation.

Another study that focused on the interpretation of recovered instant messaging data was provided by Anglano (2014). The researcher examined the WhatsApp Messenger application on an Android operating system installed in a virtual environment. As well as documenting the locations of databases and log files, these artefacts were then analysed to provide detailed descriptions of how data from disparate sources can be linked to infer meaning. While the study was based on a different mobile platform and messaging application to those used in this study, the focus on artefact analysis and interpretation is very relevant.

Just one year after Kik was released, Hoog and Strzempka (2011) briefly described Kik artefacts on iPhones installed with iOS 3.1.3 and 4.0. At the time of this study the main Kik database only had four tables (it has now grown to thirteen) and the researchers report that the user passwords were found on the devices stored unencrypted.

A more recent study was conducted by Walnycky et al. (2015). The researchers used a novel approach to acquire the related mobile data, setting up a *'man-in-the-middle'* attack to intercept messaging application traffic. A total of twenty popular instant messaging applications were used in the experiments, including Kik messenger. The results, relating to Kik, revealed that some network traffic, concerning the sharing of sketches (electronic drawings), was found to be unencrypted and could be captured in transit. The study highlighted further privacy concerns regarding a large section of the messaging applications. However, as would be expected in a high level and wide-ranging study, it did not explore Kik databases or other relevant artefacts in any detail.

There have been various articles and blogs regarding Kik iOS forensic research published online. These articles are mainly brief overviews of Kik artefacts or tutorials on how to use the commercial tools being promoted (Magnet Forensics, 2014; Timofeev et al., 2015; Manon, 2015; Sanderson, 2015). Other articles are from educational institutions discussing outputs from forensic courses (Computer & Digital Forensics Blog, 2015), and industry practitioners describing specific cases (bridgeythegeek, 2013). Of particular note is this last article, described by the author as a “work in progress”. It aimed to identify artefacts that the commercial tools did not. Although the study was incomplete and is now dated (both iOS and Kik have modified their file structures, new tables have been added to the main database, flag structures have changed and more features have been added to the application), it provided insight into the workings of Kik messenger and helped define various flags used in the databases, some of which are still applicable in the latest versions of Kik.

This research builds upon the previous efforts of researchers and practitioners and will be focusing solely on Kik messenger on iOS devices, to provide detailed, up-to-date, descriptions of Kik artefacts, locations and interpretations as well as practical guidelines on how to parse databases to provide human-readable reconstructions of user conversations.

3. Methodology

The main purpose of this study was to identify, locate and analyse artefacts modified or created by the Kik messenger application in order to answer the typical questions faced by forensic investigators in their efforts to solve crimes.

The goal of this study was achieved by conducting a set of controlled experiments using Kik messenger on iOS devices, simulating specific user scenarios: one-to-one chats, group communications, image and video exchange, etc. After each experiment, the relevant data was copied to a forensic workstation for manual analysis.

Access to the iOS file system was required to observe what files were being created and modified during usage of the Kik application. As this meant that a comprehensive manual examination was required, two of the iOS devices were jailbroken to bypass the access restrictions placed on the iOS file system by Apple.

Jailbreaking is a process that allows users to install and execute applications that have not been approved by Apple. These applications were required by the researcher to access the iOS devices with a fully interactive shell and query the file system to observe changes as they happened within the Kik application.

After the analysis revealed which Kik artefacts were of potential forensic interest, a backup of the third iOS device was performed using iTunes. This was done to ensure that the same type of artefact could be acquired by a forensic investigator without having to jailbreak the device.

The experiments were broken down into three main stages:
The **first stage** consisted of preparing the iOS devices. A factory reset of the iOS device was performed to wipe all previous contents and settings. iOS 9.02 was installed before jailbreaking two of the devices and installing the software package manager, Cydia, and various file system tools detailed in Table 1. Kik messenger was downloaded from the App Store and installed in all three iOS devices. New user accounts were created for each device.

Table 1: Devices and software used in application testing.

Device/ Software	**Use**	**Company/ Developer**	**Version**
iPad mini x 3 (MD531B/A)	Devices to be analysed	Apple	iOS 9.0.2
Kik	Application testing	Kik Interactive Inc.	9.6.0
iTunes	Create backup of iPads	Apple	12.3.2.35
Taig Jailbreak Tool	Gain access to iOS file system	Taig	2.4.1
Cydia	Software manager to install file system tools	Jay Freeman (saurik)	1.1.26
www.hexed.it	Online hex editor to analyse binary files	Jens Duttke	2015.09.02
openSSH	Secure Shell (ssh) connection to iPads	Jay Freeman (saurik)	6.7p1-13
SQLite3	View SQLite databases	sqlite.org	3.8.10.2
Python	Code scripts for data analysis and file carving	Python Software Foundation	2.7.10
Find Utilities	Indexes and searches file systems	Jay Freeman (saurik)	4.2.33-7

The **second stage** involved querying the jailbroken iOS devices to identify artefacts that were created or modified after the first step. The file systems were queried to identify newly created or modified files using the following Unix *find* command:

```
find / -type f -mmin -2 > kikFiles.txt
```

This command is broken down and explained as follows:

- `find` - search
- `/` - from the root of the filesystem
- `-type f` - for a regular file
- `-mmin -2` - that has been modified in the past two minutes
- > `kikFiles.txt` - redirect output to a text document

This query inevitably produced many false positives, as there are continuous non-related background processes also running. The false positives were reduced by repeating the process at various times, not using other applications before executing the message exchange, monitoring network activity and manual inspection and filtering of non-related data.

The relevant files were then hashed and copied over to a forensic workstation for further analysis. This step was repeated after every interaction, in order to identify the changes made to the file systems. The types of interactions included: searching for friends and groups to initiate chats, sending and receiving messages from users and groups, sending and receiving images and videos.

On occasion, for example, when trying to identify changes in message status, the analysis would take place on the iOS devices. The relevant databases would be systematically queried at source from the command line using *sqlite3*, to monitor the changing flags that indicated the progress of the message as it left the device and was delivered to the recipient.

The **third and final stage** involved manually analysing the artefacts. To begin with the files were hashed again to confirm that they had not been modified in transition. Analysis techniques included file identification, string searching, filtering, file carving, and browsing file systems. Databases were analysed with the *sqlite3* command utility and then converted, using a crafted Python script (available from author), to a spreadsheet format for easier viewing and analysis. Binary property lists (plists) were converted to XML and analysed using a text editor.

After the study was completed, and to confirm that the important artefacts identified were able to be acquired using an established method, a logical acquisition of the iOS device, that was not jailbroken, was performed using the iTunes application. A logical acquisition interacts with the device file system and can produce many important artefacts. A limitation of this method is that deleted items are not recovered. However, all artefacts referred to in this study can be acquired from the iTunes backup files.

Other established methods could also have been used to acquire the data, such as the commercial tools, Lantern by Katana Forensics, Forensic Extractor by Oxygen Forensics, and the Elcomsoft Mobile Forensic Bundle. However, iTunes was utilised as it is free application used for synchronising and backing up various Apple devices. While not a forensic data acquisition tool, this method has been leveraged by various researchers to extract iOS device artefacts (Bader and Baggili, 2010; Said et al., 2011; Hay et al., 2011; Al Mutawa et al., 2012).

If the artefacts were available from backup files, then it would be expected that commercial tools would have little difficulty in acquiring the relevant data. The iTunes application offers users the choice of storing backup files locally on a host computer, with our without encryption, or remotely on Apple servers in an iCloud account. Following the procedures established by Bader and Baggili (2010), all the important artefacts discussed in the following section, were located and identified.

4. Forensic Analysis of Kik messenger

Without any need to provide a mobile phone number or validated email address, users can immediately begin communicating with other Kik users after downloading and installing the Kik messenger application from the iTunes store. After creating a profile consisting of a unique username and optional profile picture, users can search for the username of friends or find the hashtag of public groups they wish to join. This is all that is required to begin exchanging text messages, pictures, and videos in relative anonymity.

Kik messenger uses a client-server model with Kik's servers relaying the messages between users, storing user profiles for indexing and searching, as well as authenticating users to their profiles when logging in.

The rest of this section discusses the artefacts created by the usage of the Kik messenger application, beginning with the installation locations, then focusing on contact artefacts and finally analysing message exchanges.

4.1. Kik installation files

Prior to Kik version 9.5, application code and other resources used by the code, were installed in the *Bundle* directory: `/private/var/mobile/Containers/Bundle/Application/<UUID>/Kik-Release-<x.x>.app/`
User data was stored in the *Data* directory:
`/private/var/mobile/Containers/Data/Application/<UUID>/`
Since the release of Kik version 9.5 in February 2016, there is now a third location for user data in a *Shared* directory: `/private/var/mobile/Containers/Shared/AppGroup/<UUID>/cores/private/<32-digit-hex>/`
This new directory was created to take advantage of a new feature of iOS that allows data to be shared across different applications (Apple, 2016). It contains the artefacts that would be of most interest to a forensic investigator.

These directories and files were discovered using the *find* command described in the **second stage** of Section 3 and allowed this study to focus the analysis in this location.

With the introduction of iOS 8 in September 2014, applications are no longer stored in folders of their name. Instead, the application folders are assigned a Universally Unique Identifier (UUID), that changes every time the application is updated. Consequently, locating the correct Kik folder, among the many applications the user may have installed, requires a recursive search of the *Bundle*, *Data* and *Shared* folders to locate Kik files, such as the main database, *kik.sqlite*.

The main database and other important artefact locations are listed in Table 2 and were all found in the new *Shared* location mentioned above. For Kik versions prior to 9.5, these artefacts can be found in the *Data* folder.

Table 2: Kik version (9.6.0) artefacts found in the base folder: `/private/var/mobile/Containers/Shared/AppGroup/<UUID>/cores/private/<32-digit-hex>/`

Row	Content	Folder (relative to base folder)	File
1	Kik database	.	kik.sqlite, see Table 3
2	plists containing image previews	`/attachments/`	various UUIDs
3	attachments	`/content_manager/data_cache/`	various UUIDs
4	plists containing image previews and URLs	`/content_manager/metadata_cache/`	various UUIDs
5	profile pictures	`/profpix/`	thumb_username@talk.kik.com
6	username, preferences, SHA1 of password	`/Library/Preferences/`	group.com.kik.chat.plist

Alternatively, the *kik.sqlite* database and other artefacts can be acquired from the iTunes backup files. On Mac computers, the iTunes backup files are located in:
`/Users/<username>/Library/Application Support/MobileSync/Backup/`
On Microsoft Windows PCs the backup files can be found in:
`Users\<username>\AppData\Roaming\Apple Computer\MobileSync\Backup\`

Apart from message attachments, which are stored with an UUID filename, most Kik related files on the iOS device have names and suffixes that hint at their content. However, all files that are backed up to a computer using iTunes are obfuscated using a forty-character alpha-numeric name, making it difficult to know what files relate to the Kik application among the hundreds, and sometimes thousands, of other backup files. However, there is a method available to link backup files to the original file location (Crosby, 2010).

The backup filenames are derived from a SHA1 hash of: a) the application's iOS domain, and b) part of the file's original directory on the iOS device. For example, the Kik application is part of the *AppDomainGroup* (a list of domains and files can be found in `Manifest.mbdb`, in the iTunes backup folder).

The SHA1 hash of the string: `AppDomainGroup-group.com.kik.chat-cores/private/<32-digit-hex>/kik.sqlite` produced the filename of the *kik.sqlite* database stored in the iTunes backup during testing. The filename will differ for every user due to the unique pathnames created by the variance in 32-digit hex numbers used. This is a significant change from previous versions of Kik.

As stated previously, prior to Kik version 9.5, the *kik.sqlite* database was stored in the *Data* directory on iOS devices. The SHA1 hash of the string: `'AppDomain-com.kik.chat-Documents/kik.sqlite'` is `8e281be6657d4523710d96341b6f86ba89b56df7`. Therefore, when investigating Kik versions prior to 9.5, this is the filename (for all users) for the *kik.sqlite* database stored in the iTunes backup.

Forensic software that searches for the *kik.sqlite* database in iTunes backup files using the previously static filename `8e281be6657d4523710d96341b6f86ba89b56df7`, will require an update to find the database, of newer Kik versions, by another means. Furthermore, each new version of Kik can result in variations in database structures. For example, from Kik version 8 to version 9, there are now two new tables within the *kik.sqlite* database: `ZKIKATTRIBUTION` and `ZKIKMESSAGEEXTRA`. Also, table names have changed, possibly relating to the version numbering e.g. `Z_6ADMINSINVERSE` is now named `Z_8ADMINSINVERSE`.

4.2. Analysis of contact information

Of the artefacts identified in this study and listed in Table 2, the majority of important information relating to Kik messenger usage is located within the SQLite database, *kik.sqlite*. Table 3 shows the sixteen tables that make up the database together with descriptions for each table. A thorough analysis of these tables is discussed in later sections.

Forensic investigators will often need to know with whom a suspect, or victim, has been communicating. Contact information is invaluable to any investigation. This section analysis the contacts table, `ZKIKUSER`, detailing the location of contact artefacts and then discusses how this data can be used to a) retrieve and list the current contacts, b) determine whether a contact has been blocked, c) recover deleted contacts, and d) understand group membership.

4.2.1. Retrieving contact information

The analysis of the `ZKIKUSER` table, the structure of which is shown in Table 4, revealed that it not only contained contacts that the user communicated with, but also other Kik

Table 3: Structure of the *kik.sqlite* version (9.6.0) database.

Row	Table	Description
1	`ZKIKATTACHMENT`	time stamps and associated message number, see Table 7
2	`ZKIKATTACHMENTEXTRA`	list of attachments received
3	`ZKIKATTRIBUTION`	links users to groups
4	`ZKIKCHAT`	time stamps and status of last message, see Table 5
5	`ZKIKCHATEXTRA`	last messages between contacts
6	`ZKIKMESSAGE`	time stamps and content of messages, see Table 6
7	`ZKIKMESSAGEEXTRA`	messages that have been displayed on user's screen
8	`ZKIKUSER`	list of contacts and search results for contacts, see Table 4
9	`ZKIKUSEREXTRA`	kik user's status and activity
10	`Z_4MESSAGES`	sequence of messages for each individual chat
11	`Z_8ADMINSINVERSE`	group and administrator's ID
12	`Z_8BANSINVERSE`	group ID and banned users' ID
13	`Z_8MEMBERS`	group and member's ID
14	`Z_METADATA`	plist containing hashes of table model versions
15	`Z_MODELCACHE`	data blob
16	`Z_PRIMARYKEY`	index of database tables

users suggested by the search engine when the user searched for friends using the *Find People* feature. There had been no communications with these other Kik users, which was confirmed by observing the user status field, `ZFLAGS`. Other names that appeared in the `ZKIKUSER` table were fellow members of public groups that had no direct contact with the user.

There also appeared non-human users – *bots*, which are software programs used to chat with Kik users using pattern matching techniques to appear human. These *bots* are run by Kik's administrators, as well as companies who market their brands through a feature called *Promoted Chats*.

Within the `ZKIKUSER` table there are various fields used for names or identifiers. One of these fields, `ZUSERNAME`, is the unique username that is created during registration of a new Kik account. This is the only piece of information that Kik uses to identify an account and it cannot be changed by the user. However, users can change their display name (`ZDISPLAYNAME`) whenever they wish to. Both username and display names are visible to other users.

The unique user name is used to create the Jabber ID, `ZJID`. This relates to the original Extensible Messaging and Presence Protocol (XMPP), which is the basis of Kik's, and many other messaging applications, communication architecture.

Public and private groups are given a thirteen-digit number appended with `_g@groups.-kik.com` for their Jabber ID. Public groups have the `ZGROUPTAG` field populated in the form of a hashtag. The hashtag is the name of the group and it is what is used by people to search and join groups.

Finally, each user and group are also assigned a number in the `ZEXTRA` field. The number is based on when the user or group first appeared in the user's database; the higher the number, the more recent the user or group was added to the database. This number is used in tables throughout the database, often in the `ZUSER` field, and can be used to link data together from the various tables.

With so many contacts appearing within the `ZKIKUSER` table, it is important to be able to

Table 4: ZKIKUSER version (9.6.0) database table.

Row	Field name	Description
1	Z_PK	primary key
2	Z_ENT	relates to Z_PRIMARYKEY table; 8 = ZKIKUSER table
3	Z_OPT	unknown - integer accumulates with messages and status changes
4	ZADDRESSBOOKID	always 0
5	ZFLAGS	contact status
6	ZINTERNALID	always 0
7	ZPRESENCE	always 1
8	ZTYPE	always 1
9	ZATTRIBUTION	links to the ZKIKATTRIBUTION table
10	ZCHATUSER	relates to ZKIKCHAT table
11	ZEXTRA	local ID of user
12	ZLASTMESSAGE	blank
13	ZDISPLAYNAME	the current name being used.
14	ZDISPLAYNAMEASCII	as above, in ASCII
15	ZEMAIL	blank
16	ZFIRSTNAME	chosen first name
17	ZGROUPTAG	hashtag name of group (deprecated)
18	ZJID	Jabber ID in the format of <ZUSERNAME>_<abc>.@talk.kik.com.
19	ZLASTNAME	chosen last name
20	ZPPTIMESTAMP	13 digit Unix epoch time (UTC) of when the contact uploaded their profile picture
21	ZPPURL	web URL location of the contact's profile picture
22	ZSTATUS	blank
23	ZUSERNAME	the unique name that the user has registered with Kik.
24	ZCONTENTLINKSPROTODATA	web links

differentiate between contacts that the user has communicated with and other Kik users that are of no interest to an investigation. The relationship status of other Kik users is defined by the ZFLAGS field. Numeric codes are used to represent this relationship. Experiments undertaken appear to show the following codes defined as:

- 2 – unknown
- 8,14 – groups user has left
- 9,11 – groups user is a member of
- 128 - 132 – Kik *Promoted Chats/bots* (marketing feature)
- 256 – group members and search engine suggestions, no connections.
- 257 – other users the account holder has communicated with
- 258 – Kik account user
- 260 – unknown
- 288 - 289 – blocked users
- 385 – *kikteam* administrators and connected *Promoted Chats*
- 417 – previous *Promoted Chats*, inactive

The ZFLAGS codes only relate to the relationship status at the time of data acquisition and there appears to be no way of discovering previous relationship statuses.

The profile picture a user chooses could be a clue to their identity. Whether it is a self portrait, an identifiable background object or location, or a favourite image previously used on other accounts, these artefacts can be of evidentiary value. Table 2, row no. 5, details where an investigator can locate the thumbnail profile images of Kik contacts. A further source of profile images was discovered in the ZPPURL field. This provided a uniform resource locator (URL) pointing to the Internet location of the contact's profile picture. Entering the URL into a web browser and appending /orig.jpg or /thumb.jpg retrieved a copy of the image. The time a contact last updated their profile picture was revealed in the ZPPTIMESTAMP field, which is written in a thirteen-digit Unix epoch time format, reflecting Coordinated Universal Time (UTC).

The ZKIKCHAT table, shown in Table 5, stored details of other users that were either blocked or had been involved in message exchanges with the user. The interactions that involved message exchanges were indicated by the ZLASTMESSAGE and ZDATEUPDATED field being populated. The interactions that had no messages or time stamps appear to be other users that had been blocked before any chats were initiated. The Z_OPT field relates to the volume of activity between users. It was noticeable that the greater the volume of messages exchanged between the user or group, the greater the Z_OPT number. Therefore, this data correlates with whom the user had communicated most.

Table 5: Structure of the ZKIKCHAT version (9.6.0) table.

Row	Field name	Description
1	Z_PK	primary key
2	Z_ENT	from Z_PRIMARYKEY table, 4 = ZKIKCHAT
3	Z_OPT	linked to volume of interactions
4	ZFLAGS	unknown
5	ZDRAFTMESSAGE	blank
6	ZEXTRA	sequence of interactions
7	ZLASTMESSAGE	last message with user or group
8	ZUSER	local user ID
9	ZDATEUPDATED	Mac Absolute time stamp (UTC) of last message

4.2.2. Dealing with blocked contacts

Kik provides a facility to block unwanted contact from other users. Any subsequent messages from a blocked person will be hidden from the user. However, experiments showed that messages from a blocked user were still delivered to the device and could be found in the ZKIKMESSAGE table in the *kik.sqlite* database. Although the Kik website states that all chats with the blocked person will be deleted, this only applied to what was displayed to the user. Previous chat messages were still present in the ZKIKMESSAGE table and were used to reinstate the conversation exchange when the other Kik user was unblocked. This included messages previously not displayed while the other Kik user was blocked. Attachments from blocked users were still present in the locations shown in Table 2, row no. 3. The other user's status, shown in the ZFLAGS field, changed from *257* to *256* when the user was blocked, and reverted back to *257* when unblocked. No evidence could be found as to when a contact was blocked or unblocked.

4.2.3. Recovering deleted contacts

Kik provides users with the option to delete contacts from their chat lists. Selecting this option removes the contact, and any previous conversations with that contact, from the chat list display. From the perspective of the Kik user, the *'deleted'* contact and chats are no longer available.

From the perspective of a forensic investigator, the *'deleted'* contact and chats could still be recoverable from within the *kik.sqlite* database. Experiments undertaken showed that deleting a contact had no noticeable effect on the `ZKIKUSER` table (which stores contacts' details) and no noticeable effect on the `ZKIKMESSAGE` table (which stores the content and details of previous chats). How long this data remains within the database is dependent upon the volume of messages the user subsequently sends and receives, as the data will eventually be overwritten with new data. This issue is discussed further in Section 4.3.5, *Deleted messages*.

4.2.4. Understanding group membership

Kik messenger provides the facility for users to chat with each other as part of a group. The groups can have up to fifty members and are either private, where entry is by invitation, or public, open to anyone.

Analysis of the *kik.sqlite* database found four tables concerned exclusively with Kik groups, Table 3, rows 3 and 11-13. The `Z_8MEMBERS` table listed number codes relating to each group the user was currently a member of, along with the number codes of other group members. These number codes refer to the `ZEXTRA` field in the `ZKIKUSER` table, which provides group names and the unique usernames of members. Group administrators could be identified from the `Z_8ADMINSINVERSE` table and a list of banned users was found in the `Z_8BANSINVERSE` table.

Group members were added to the `ZKIKMESSAGE` table when they joined, left or posted messages to the group. Despite being members of the same group, the `ZFLAGS` status of other group members remained at *256*, unless the user engaged in a one-to-one chat with the other member.

Group administrative messages notified group members when new users joined groups, users were blocked, and group profile picture was changed. If the original group message is still in the database, the time stamp of this message can be used to determine when a user joined the group, as the original message comes from the Kik administration account stating the user has joined. Other than the initial *join* message, there was no other evidence found that could relate to when an individual was added to a group.

When the user sent a message to a group, the group's `ZEXTRA` number from the `ZKIKUSER` table was used to indicate the recipient. However, messages received from a group appear as they would from an individual, i.e. it is the individual's `ZEXTRA` number that appears in the user's database, not the group number. This can make it difficult for an investigator to reconstruct conversations, as it is not clear on first viewing, if a message was received as a direct message from the other user, or if it came via the group and therefore visible to all other group members. This issue is discussed further in Section 4.3.2, *Reconstructing the chat history*.

As well as proactively joining a group, users could be added to a group by other Kik users. For this to happen there would need to have been previous communication between the two users, i.e. the `ZFLAGS` status of the user must be *257*; a user cannot be added to a group unless they have previously chatted with them. Once the user was added to a group the `ZKIKMESSAGE` table contained a message stating the user has been added. Subsequent messages and attachments were downloaded into the user's database even if the user had not accessed the group. It should be noted that the user may not know the nature and content of the group to which they have been added, and they may have not actually accessed or viewed any messages from the group.

This section dealt with the question as to whom a user has been communicating with. More information is provided in relation to communications, including content and timing in the next section.

4.3. Analysis of exchanged messages

As well as knowing with whom a user has been communicating, investigators will want to know when the communications took place and the content of the messages. Furthermore, an investigator will want to know if any attachments were exchanged and where these could be found on the devices.

The following section, *Structure of the message exchange table*, 4.3.1, starts by revealing where exactly in the *kik.sqlite* database, message content can be found. It then explains how to recreate a message exchange timeline in *Reconstructing the chat history*, 4.3.2. How to identify and extract attachments is discussed in *Message Attachments*, 4.3.3. Messages pass through various stages en route from sender to receiver and these stages are clarified in *Determining the status of a message* 4.3.4. Finally, *Deleted messages*, 4.3.5, discusses the forensic implications of messages the user has attempted to delete.

4.3.1. Structure of the message exchange table.

Message content, as well as metadata, is stored in the `ZKIKMESSAGE` table, shown in Table 3, row no. 6, within the *kik.sqlite* database. Table 6 shows the nineteen fields that make up the table and, where known, provides a definition. The main fields of interest include `ZBODY`, which contains the actual message text, the `ZFLAGS` field, which indicates if the message contains attachments, and the time stamp fields, `ZTIMESTAMP` and `ZRECEIVEDTIMESTAMP`.

4.3.2. Reconstructing the chat history

To construct a meaningful chat timeline the information sought from the *kik.sqlite* database consisted of:

- what was said? - message content
- who said it and who did they say it to? - sender/receiver
- when were the messages sent? - message time stamps
- the correct sequence of the conversation - chronology

Data from fields in four different tables was threaded together to produce the chat history. Figure 1 illustrates the connections between the tables and the fields required to reconstruct the chat. The following process shows how a forensic investigator would use this information to isolate a specific chat history:

Table 6: Structure of the ZKIKMESSAGE version (9.6.0) table.

Row	Field name	Description
1	Z_PK	primary key
2	Z_ENT	from Z_PRIMARYKEY 6 = ZKIKMESSAGE
3	Z_OPT	unknown - No's. 1 to 9 displayed
4	ZFLAGS	4 = attachment, 0 = all other messages
5	ZINTERNALID	chat sequence
6	ZSTATE	delivery status - see Section 4.3.4
7	ZSYSTEMSTATE	12 = sent messages, 0 = all other messages
8	ZTYPE	1 = received, 2 = sent, 3 = housekeeping messages to group admin, 4 = housekeeping messages to group
9	ZCHATEXTRA	links to ZKIKCHATEXTRA table
10	ZDRAFTMESSAGECHAT	blank
11	ZEXTRA	links to ZKIKMESSAGEEXTRA table
12	ZLASTMESSAGECHAT	relates to last message from user displayed on Kik app home screen
13	ZLASTMESSAGEUSER	blank
14	ZUSER	ZEXTRA ID from ZKIKUSER table
15	ZRECEIVEDTIMESTAMP	Mac Absolute time (UTC) of received message
16	ZTIMESTAMP	Mac Absolute time (UTC) of sent message
17	ZBODY	message content
18	ZSTANZAID	UUIDs
19	ZRENDERINSTRUCTIONSET	blank

1. Starting with the username to be investigated (ZDISPLAYNAME), a query of the ZKIKUSER table would provide the ZEXTRA number relating to this user.
2. Locating this number within the ZUSER field of the ZKIKCHAT table, the investigator would note the ZEXTRA number, which is the conversation chat number.
3. This number links to the Z_4CHAT field in the Z_4MESSAGES table and will provide all message numbers that relate to that chat number in the Z_6MESSAGES field.
4. Finally, the message numbers can then be used to select the message content from the ZKIKMESSAGE table, along with the time of receipt and message type (sent, received, etc).

Combining these details from the various tables isolated the conversations that had taken place between the user and specific Kik contacts or groups. It also clarified if messages were sent as direct messages to an individual or as part of a group discussion. Figure 2 shows the *sqlite3* query which produced a more human-readable format.

The command shown in Figure 2 has been broken down to provide an explanation:

- Line 1 - open the database with SQLite3 and display in columns with headings
- Line 2 - select the following: ZBODY but rename it to 'Message'
- Line 3 - ZRECEIVEDTIMESTAMP and convert it to local time in a human-readable format, renaming it 'Date'
- Lines 4-9 - ZTYPE renamed as 'Type' and change the values: 1 = rcvd, 2 = sent, 3 & 4 are group related messages
- Line 10 - ZDISPLAYNAME but rename it 'User'
- Line 11 - Z_6MESSAGES but rename it 'Message No.'
- Line 12 - from ZKIKMESSAGE table...
- Line 13 - ...join to the ZKIKUSER table where ZEXTRA and ZUSER match

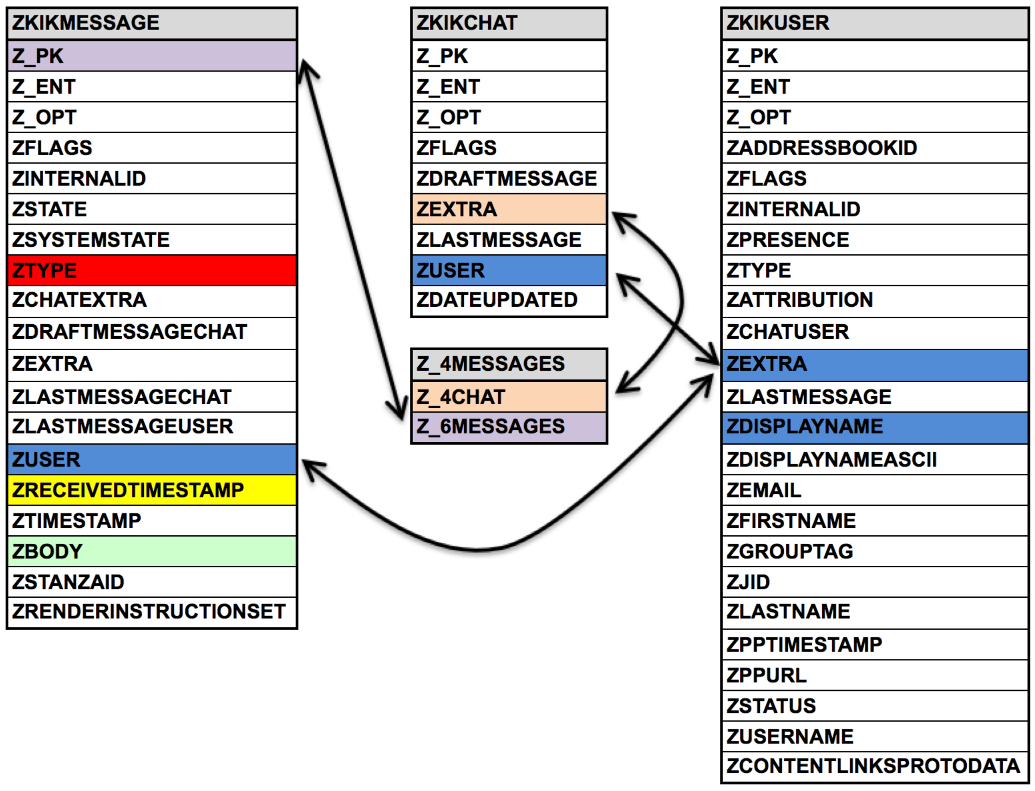


Figure 1: Data taken from four tables to produce the chat history. The arrows link corresponding fields.

- Line 14 - and join to the `Z_4MESSAGES` table where `Z_6MESSAGES` and `Z_PK` match, displaying the messages that relate to a chat number (1 in this instance)

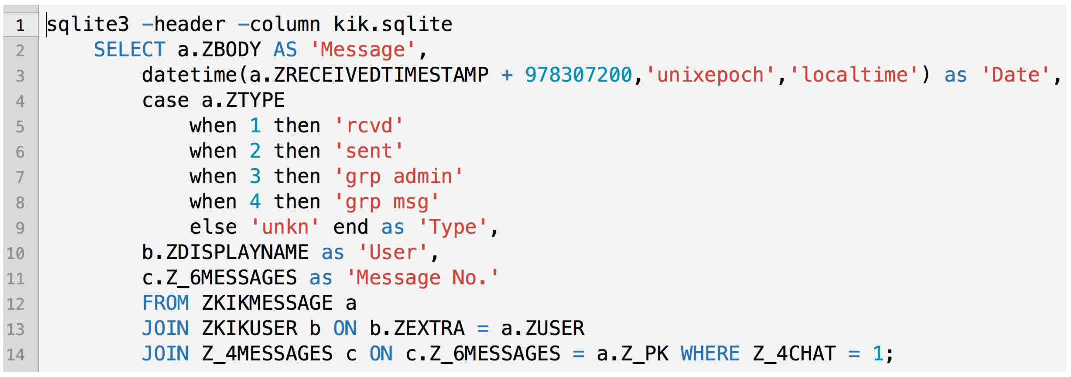
Figure 3 shows the output of the *sqlite3* query presented it in a more informative format. The output is colour-coded with the field colours of Figure 1, to link the query output back to the original table location of the data.

This method of threading together conversations would also apply to version 8 of Kik. However, the field and table names would need to be adjusted to suit i.e. `Z_4MESSAGES`, `Z_6MESSAGES` and `Z_4CHAT` were previously named `Z_3MESSAGES`, `Z_5MESSAGES` and `Z_3CHAT` respectively.

4.3.3. Message attachments

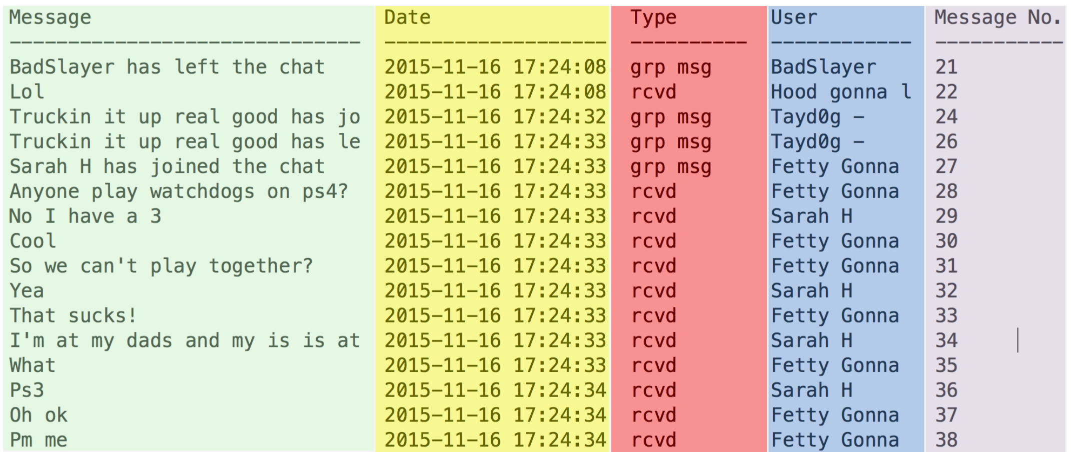
This section addresses the questions an investigator would have relating to a Kik user's exchange of pictures and other attachments. As well as exchanging text messages, Kik allows its users to send picture and video files. There is also the facility to share web content - stickers, viral videos, sketches, etc. This section will discuss the process and creation of new artefacts when sending and receiving attachments, the structure of the attachments database table and the consequences of users deleting messages containing attachments.

When a Kik user sends an image or video to another contact or a group, the file is uploaded to Kik servers at `https://platform.kik.com/content/files/<UUID>` and a copy



```sql
sqlite3 -header -column kik.sqlite
    SELECT a.ZBODY AS 'Message',
        datetime(a.ZRECEIVEDTIMESTAMP + 978307200,'unixepoch','localtime') as 'Date',
        case a.ZTYPE
            when 1 then 'rcvd'
            when 2 then 'sent'
            when 3 then 'grp admin'
            when 4 then 'grp msg'
            else 'unkn' end as 'Type',
        b.ZDISPLAYNAME as 'User',
        c.Z_6MESSAGES as 'Message No.'
        FROM ZKIKMESSAGE a
        JOIN ZKIKUSER b ON b.ZEXTRA = a.ZUSER
        JOIN Z_4MESSAGES c ON c.Z_6MESSAGES = a.Z_PK WHERE Z_4CHAT = 1;
```

Figure 2: The SQLite query used to parse the chat history.



```
Message                         Date                 Type        User          Message No.
------------------------------  -------------------  ----------  ------------  -----------
BadSlayer has left the chat     2015-11-16 17:24:08  grp msg     BadSlayer     21
Lol                             2015-11-16 17:24:08  rcvd        Hood gonna l  22
Truckin it up real good has jo  2015-11-16 17:24:32  grp msg     Tayd0g -      24
Truckin it up real good has le  2015-11-16 17:24:33  grp msg     Tayd0g -      26
Sarah H has joined the chat     2015-11-16 17:24:33  grp msg     Fetty Gonna   27
Anyone play watchdogs on ps4?   2015-11-16 17:24:33  rcvd        Fetty Gonna   28
No I have a 3                   2015-11-16 17:24:33  rcvd        Sarah H       29
Cool                            2015-11-16 17:24:33  rcvd        Fetty Gonna   30
So we can't play together?      2015-11-16 17:24:33  rcvd        Fetty Gonna   31
Yea                             2015-11-16 17:24:33  rcvd        Sarah H       32
That sucks!                     2015-11-16 17:24:33  rcvd        Fetty Gonna   33
I'm at my dads and my is is at  2015-11-16 17:24:33  rcvd        Sarah H       34
What                            2015-11-16 17:24:33  rcvd        Fetty Gonna   35
Ps3                             2015-11-16 17:24:34  rcvd        Sarah H       36
Oh ok                           2015-11-16 17:24:34  rcvd        Fetty Gonna   37
Pm me                           2015-11-16 17:24:34  rcvd        Fetty Gonna   38
```

Figure 3: The output of the SQLite query.

of the attachment is stored on the device. A plist containing a preview version of the attachment is also created and stored. A further plist containing a link to the URL is also created. The locations of these files are shown in Table 2, rows 2, 3 and 4.

On the recipient side, the user is notified of a new message by a push notification, if this has been permitted in the iOS security settings. When the recipient opens the Kik application, all chats are updated and attachments are downloaded to the same location mentioned above for sent files; Table 2, rows 2, 3 and 4. The recipient does not have to view the files for them to be downloaded onto their device. This happens automatically when the Kik application is opened.

When accessing specific chats or groups, the preview or thumbnail version of the attachment appears on the chat timeline. The user can click on the file to view the full version on the screen. If the recipient chooses to save the attachment, copies are made and are located in the usual iOS media locations, e.g.

- `/private/var/mobile/Media/DCIM/100APPLE/`
- `/private/var/mobile/Media/PhotoData/MISC/`
- `/private/var/mobile/Media/PhotoData/Thumbnails/`

- /private/var/mobile/Media/PhotoStreamsData/8265296843/100APPLE/

Experiments showed that the process of saving an attachment, modified the Exif data to include a thumbnail image, resulting in a hash value mismatch with the original image stored in .../content_manager/datacache/. The images and videos exchanged during the experiments were analysed to check for identifiable metadata, such as geolocation tags. However, none were found.

With regards to the URLs embedded within plists, the Kik application used a key or token for authentication purposes to download the attachments. The full URL and token could be viewed within the plists referred to in Table 2, row no. 4. After extracting the URL from the plist, pictures and videos were able to be downloaded directly from the server using a web browser, without needing to use the Kik application.

Information relating to messages with attachments were found in the ZKIKATTACHMENT table, Table 7 within the kik.sqlite database. The attachments were identified by a UUID number in the ZCONTENT field, which was then used to locate the actual file from the folder referred to in Table 2, row no. 3. In both the ZKIKATTACHMENT and ZKIKMESSAGE tables, the presence of an attachment, regardless of the type of attachment, was indicated in the ZFLAGS field with the number *4*.

Table 7: Structure of the ZKIKATTACHMENT version (9.6.0) table.

Row	Field name	Description
1	Z_PK	primary key
2	Z_ENT	from Z_PRIMARYKEY table, 1 = ZKIKATTACHMENT
3	Z_OPT	1 = attachment present, 2 = deleted
4	ZFLAGS	4 = attachment
5	ZINTERNALID	always 0
6	ZRETRYCOUNT	always 0
7	ZSTATE	always 0
8	ZTYPE	unknown, 6,7 displayed
9	ZEXTRA	sequenced numbers are attachments received, blanks are attachments sent
10	ZMESSAGE	message no. from ZKIKMESSAGE table
11	ZLASTACCESSTIMESTAMP	blanks
12	ZTIMESTAMP	Mac Absolute time (UTC) attachment uploaded (sent) or downloaded (received)
13	ZCONTENT	UUID of attachment

Message attachments that were deleted from the Kik application were still referenced in the ZKIKATTACHMENT table. An indicator that the user deleted the message is the Z_OPT field being populated with the number *2*. There was also the absence of a ZMESSAGE number referring to the message from the ZKIKMESSAGE table. These can be taken as indicators that the user has attempted to delete the message containing the attachment.

While this action will have been successful in removing the message and attachment from the Kik user interface, it fails to remove the attachment from the iOS device. Indeed, the actual attachment files remained intact in both the device and on the Kik servers. In this study, pictures deleted from the Kik application were still available via the URL on Kik servers for a further four weeks, and for a further eight weeks from the iOS device. Property lists, located in Table 2, row 2, embedded with preview versions of the original image, were still recoverable from the iOS device and were still being backed up by iTunes three months after the original image had been deleted on the application.

4.3.4. Determining the status of a message

As stated previously, messages are not sent directly between users. Rather, the messages are uploaded to Kik servers before being forwarded to the recipient. If the recipient's iOS device is locked, a push notification is sent from Kik informing them of the message, which is only then delivered to the device after the recipient opens the Kik application. This explains why there can sometimes be a significant time difference between when the message was sent (`ZTIMESTAMP`), and the time the recipient received the message (`ZRECEIVEDTIMESTAMP`). The message status depends on various factors: Has the message been delivered to the Kik server? Has the recipient been notified of a new message? Was the message delivered immediately? Has the message been read?

From experiments undertaken the following `ZSTATE` codes from `ZKIKMESSAGE` table were discovered and defined as follows:

- Received messages
  - 0 – message unread
  - 16 – message read
- Sent messages
  - 0 – administrative messages regarding groups
  - 2 – message not sent from device
  - 6 – sent to Kik server
  - 10 – unknown
  - 14 – delivered immediately, solid D - push-notification sent
  - 22 – unknown
  - 30 – read after immediate delivery
  - 38 – not a member of the group
  - 66 – delay in reaching Kik servers
  - 70 – faded D - push notification sent
  - 78 – delivered after being offline
  - 90 – unknown
  - 94 – read after being offline

4.3.5. Deleted messages

As part of the experiments undertaken a selection of individual messages were deleted using the Kik application. Subsequent analysis of the *kik.sqlite* database found that the messages were also deleted from the `ZKIKMESSAGE` table, as would be expected. However, as mentioned previously, the deleted attachments were still referenced in the `ZKIKATTACHMENT` table and the files were still stored in the folder shown in Table 2, row no. 3. Furthermore, when an entire conversation was deleted, the messages were no longer displayed on the application, but surprisingly, the entire message exchange remained in the `ZKIKMESSAGE` table. All messages and attachments remained in place. When new conversations were initiated with the same user, it appeared as a new chat on the application (no historic conversations were displayed), but the previous chats all remained within the `ZKIKMESSAGE` table.

The Kik website states that users can view the last 1,000 recent messages. Recent being defined as the last forty-eight hours. For older messages the last 500 will be viewable (Kik, 2015d). What a forensic investigator is able to recover from a device, will therefore be dependent on how much activity there has been. In a busy chat where many messages are exchanged, the older messages will be deleted as newer messages appear. However, in quieter chats, messages may still be recoverable from the database long after they have been deleted from the screen by the user. For example, during this study, messages that had been deleted by the user were still available from the device three months after the messages had been sent and one month after they had been deleted.

5. Conclusions and future work

Messaging applications on mobile platforms can provide a significant volume of information to forensic investigators. Data created by user interactions with the applications are often stored in plaintext, but interpreting this data is not always straightforward due to the usage of undocumented, and sometimes, obscure naming conventions.

This study focused on the recovery of artefacts relating to the use of Kik messenger on iOS devices. It documented where artefacts of interest could be found on the device, as well as backup files on personal computers. Many code meanings could only be deciphered through multiple experiments involving sent and received messages from both individuals and groups, varying message types and attachments, altering user, message, and device states, e.g., offline, online, blocked, removed, deleted. This data was then analysed to answer the typical questions that would arise during a forensic investigation:

1. Who has the user been communicating with and when?
2. What was the content of the communications?
3. What attachments were exchanged and where can they be found?

By documenting the location and type of Kik messenger artefacts, on both the iOS devices and the iTunes backups, forensic investigators will be able to determine who has been communicating with whom and when, answering the first question. Identifying the actual person behind the Kik username may prove more difficult for investigators. Unlike other instant messaging registration processes, there is little effort shown by Kik to verify a user's identity; there is no requirement to link a mobile phone number to the account and email verification is not mandatory. There are no relevant log files stored on the devices and the descriptive metadata embedded within images and videos is removed. While this offers a certain level of privacy for Kik users, it also makes it difficult for law enforcement to identify those who would use the application in the commission of a crime. It may be possible, however, to identify users from their profile pictures, posted images, or the contents of their communications. These artefacts can be found in abundance within the iOS devices and the iTunes backup files.

The content of communications was put into context in *Reconstructing the chat history* 4.3.2, answering the second question. While the third question, relating to messages attachments, was answered through a detailed explanation of the creation of new artefacts that showed, not only where attachments are stored on devices, but also the various traces and copies of attachments that can also be recovered from the device, backup files and from Kik servers.

This study can help forensic investigators interpret Kik messenger artefacts retrieved from iOS devices. It can also be of value to forensic extraction tool developers in building software applications that can accurately retrieve all relevant data to help reconstruct past communications. To this end, a plugin for the *plaso* framework, which is used in various forensic tools such as *log2timeline*, has been developed. The plugin is available to download as part of the *plaso* framework from `https://github.com/log2timeline/plaso`.

Future work should involve Kik messenger analysis on other platforms including Android, Windows, and Amazon. New messaging applications emerge regularly and can gain huge popularity in a short space of time. Continued research into these communication applications is essential to assist law enforcement in their investigations.

References

Al Mutawa, N., Baggili, I., Marrington, A., 2012. Forensic analysis of social networking applications on mobile devices. Digital Investigation 9 (SUPPL.), 24–33.

Alvarez, L., September 2013. Girl's Suicide Points to Rise in Apps Used by Cyberbullies. [Accessed: 2015-9-15].
URL http://www.nytimes.com/2013/09/14/us/suicide-of-girl-after-bullying-raises-worries-on-web-sites.html

Anglano, C., Sep 2014. Forensic analysis of whatsapp messenger on android smartphones. Digital Investigation 11 (3), 201–213.
URL http://www.sciencedirect.com/science/article/pii/S1742287614000437

Apple, 2016. App Extension Programming Guide: App Extensions Increase Your Impact. [Accessed: 2016-02-25].
URL https://developer.apple.com/library/ios/documentation/General/Conceptual/ExtensibilityPG/index.html

Bader, M., Baggili, I., 2010. iphone 3gs forensics : Logical analysis using apple itunes backup utility. Small Scale Digital Device Forensics Journal 4 (1), 1–15.
URL http://securitylearn.net/wp-content/uploads/iOSResources/iPhone3GSForensicsLogicalanalysisusingAppleiTunesBackupUtility.pdf

bridgeythegeek, June 2013. Artefacts of Kik Messenger on iOS. [Accessed: 2015-9-15].
URL http://www.scribd.com/doc/145278610/Artefacts-of-Kik-Messenger-on-iOS

Computer & Digital Forensics Blog, March 2015. Mobile Device Apps: Kik. [Accessed: 2015-9-15].
URL http://computerforensicsblog.champlain.edu/2015/03/02/mobile-device-apps-kik/

Crosby, A., March 2010. iPhone Forensics, sans iPhone. [Keynote Presentation],[Accessed: 2015-10-02].
URL http://blog.uptill3.com/static/iphone_forensics.pdf

Federal Bureau of Investigation, September 2015. Two Rochester Men Arrested on Child Exploitation Charges. [Accessed: 2015-9-15].
URL https://www.fbi.gov/buffalo/press-releases/2015/two-rochester-men-arrested-on-child-exploitation-charges

Godfrey, M., March 2013. Pedophiles coercing kids using phone app. [Accessed: 2015-10-30].
URL http://www.smh.com.au//breaking-news-national/pedophiles-coercing-kids-using-phone-app-20130327-2gu3a.html

Hay, A., Krill, D., Kuhar, B., Peterson, G., 2011. Evaluating digital forensic options for the apple ipad. In: Peterson, G., Shenoi, S. (Eds.), Advances in Digital Forensics VII. Vol. 361 of IFIP Advances in Information and Communication Technology. Springer Berlin Heidelberg, pp. 257–273.
URL http://dx.doi.org/10.1007/978-3-642-24212-0_20

Hoog, A., Strzempka, K., 2011. iPhone and iOS forensics: Investigation, analysis and mobile security for Apple iPhone, iPad and iOS devices. Elsevier.

Husain, M., Sridhar, R., 2010. iforensics: Forensic analysis of instant messaging on smart phones. In: Goel, S. (Ed.), Digital Forensics and Cyber Crime. Vol. 31 of Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, pp. 9–18.
URL http://dx.doi.org/10.1007/978-3-642-11534-9_2

Husain, M. I., Baggili, I., Sridhar, R., 2011. A Simple Cost-Effective Framework for iPhone Forensic Analysis. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST 53, 27–37.

Kik, 2015a. Kik Guide For Law Enforcement. [Accessed: 2015-9-15].
URL https://kiklawenforcement.zendesk.com/hc/en-us/article_attachments/201789055/TS1.1_Kik_s_Guide_for_Law_Enforcement_November_13__2014.pdf

Kik, 2015b. Kik Our Beliefs. [Accessed: 2015-9-15].
URL http://www.kik.com/about/

Kik, 2015c. Kik Terms of Service. [Accessed: 2015-9-15].
URL www.kik.com/assets/Uploads/Kik-Terms-of-Service-Posted-June-29-2015.pdf

Kik, 2015d. Top faqs. [Accessed: 2015-10-07].
URL https://kikinteractive.zendesk.com/entries/87183183-Can-I-save-my-Kik-messages

Larson, S., Oct 2015. How Kik became the king of sketchy messaging apps. [Accessed: 2015-10-30].
URL http://www.dailydot.com/technology/kik-drugs-sex-messaging-apps/

Levinson, A., Stackpole, B., Johnson, D., 2011. Third Party Application Forensics on Apple Mobile Devices. Proceedings of the Annual Hawaii International Conference on System Sciences, 1–9.

Magnet Forensics, 2014. The Rise of Mobile Chat Apps: Recovering Evidence From Kik Messenger, WhatsApp & BBM. Tech. rep., Magnet Forensics Inc., 156 Columbia St W, Waterloo, ON N2L 3L3, Canada, [Accessed: 2015-9-15].
URL https://www.magnetforensics.com/mobile-forensics/the-rise-of-mobile-chat-apps-recovering-evidence-from-kik-messenger-whatsapp-bbm/

Manon, A., July 2015. Kik Messenger Forensics. [Accessed: 2015-9-15].
URL http://www.xploreforensics.com/blog/kik-messenger-forensics.html

Ofcom, August 2015. The Communications Market 2015. [Accessed: 2015-9-15].
URL http://stakeholders.ofcom.org.uk/market-data-research/market-data/communications-market-reports/cmr15/

Piccinelli, M., Gubian, P., 2011. Exploring the iPhone Backup Made by iTunes. Journal of Digital Forensics, Security and Law 6 (3), 31–63.

Said, H., Yousif, A., Humaid, H., 2011. Iphone forensics techniques and crime investigation. Proceedings of the 2011 International Conference and Workshop on the Current Trends in Information Technology, CTIT'11, 120–125.

Sanderson, P., May 2015. Kik binary plist decoder. [Accessed: 2015-9-15].
URL http://sandersonforensics.com/forum/content.php?227-Kik-binary-plist-decoder

Sgaras, C., Kechadi, M.-T., Le-Khac, N.-A., 2015. Forensics acquisition and analysis of instant messaging and voip applications. In: Garain, U., Shafait, F. (Eds.), Computational Forensics. Vol. 8915 of Lecture Notes in Computer Science. Springer International Publishing, pp. 188–199.
URL http://dx.doi.org/10.1007/978-3-319-20125-2_16

Timofeev, N., Afonin, O., Gubanov, Y., Makeev, D., 2015. Kik Messenger Forensics. [Accessed: 2015-9-15].
URL http://ru.belkasoft.com/en/kik-forensics

Tso, Y.-C., Wang, S.-J., Huang, C.-T., Wang, W.-J., 2012. iPhone Social Networking for Evidence Investigations Using iTunes Forensics. In: Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication. ICUIMC '12. ACM, New York, NY, USA, pp. 62:1–62:7.
URL http://doi.acm.org/10.1145/2184751.2184827

Van Vleck, T., Jan 2012. Electronic Mail and Text Messaging in CTSS, 1965-1973. Annals of the History of Computing, IEEE 34 (1), 4–6.

Walnycky, D., Baggili, I., Marrington, A., Moore, J., Breitinger, F., 2015. Network and device forensic analysis of android social-messaging applications. Digital Investigation 14 (AUGUST), S77–S84.
URL http://linkinghub.elsevier.com/retrieve/pii/S1742287615000547

Zauzmer, J., September 2014. Va. church volunteer charged with sending sexually explicit messages to teenager. [Accessed: 2015-9-15].
URL https://www.washingtonpost.com/local/crime/va-youth-pastor-charged-with-sending-sexually-explicit-messages-to-teenager/2014/09/22/f127c498-4284-11e4-9a15-137aa0153527_story.html