Forensic Analysis of Secure Ephemeral Messaging Applications on Android Platforms

M A Hannan Bin Azhar and Thomas Edward Allen Barton

Computing, Digital Forensics and Cybersecurity
Canterbury Christ Church University
Canterbury, United Kingdom
{hannan.azhar,tb1150}@canterbury.ac.uk

Abstract. Secure messaging applications have been used for the purposes of major crime, creating the need for forensic research into the area. This paper forensically analyses two secure messaging applications, Wickr and Telegram, to recover artefacts from and then to compare them to reveal the differences between the applications. The artefacts were created on Android platforms by using the secure features of the applications, such as ephemeral messaging, the channel function and encrypted conversations. The results of the experiments documented in this paper give insight into the organisation of the data structures by both Wickr and Telegram, as well as the exploration of mobile digital forensics techniques to recover artefacts removed by the ephemeral functions.

Keywords: Wickr, Telegram, Ephemeral Data, Secure Messaging Application, Android, Forensic Analysis, Mobile Forensics.

1 Introduction

Wickr and Telegram are Secure Messaging Applications (SMAs), with an estimated 1-5 and 100-500 million installs on Android Platforms respectively [1]. The popularity of SMAs, while a great tool for change and communication for many, have also seen them used for major crimes. In 2015, there was a huge spike in acts of terrorism in both European countries and around the world [2, 3]. While terror attacks are not new, some recent events have seen a new trend: the usage of social media and networking by terrorist groups, for the proliferation of their causes [4, 5, 6]. Whether used for co-ordinations, communications or recruitments, the need for forensic analysis and a greater understanding of the SMAs has never been more urgent.

The capitalization of mobile applications by markets such as Google Play [1] and the iTunes App Store [7] has driven the advent of new functionality for all applications, including SMAs. In this age of information, the demand for secure methods of communication has increased. SMAs ensure privacy of communications, which regardless of the implications, present a challenge to the digital forensic investigators because of the intentional obfuscation and removal of artefacts, which hampers investigations, often by increasing the amount of time and money invested, eventually reducing the likelihood of a successful completion [8].

This paper will forensically analyse two secure messaging applications, Wickr [9] and Telegram [10]. These applications were chosen with their features in mind. Both apps have an ephemeral messaging function. Ephemeral messaging is a relatively novel and popular way of social networking. Ephemeral messaging applications (EMA) use transient data, data that are permitted to exist for a limited amount of time before they become inaccessible, by obfuscation or deletion. An example of an EMA that rose to huge popularity among the eighteen to thirty-four year old age range (which happens to be statistically the largest demographic for smartphone ownership in the United States [11]) was SnapChat [12], an application that allowed users to send photos and images that would be deleted after opening. SnapChat brought EMAs into general knowledge and since then, many other applications have adopted the ephemeral feature, including SMAs.

The remainder of the paper is organised as follows: Section 2 describes related work, similar forensic research into various types of secure and non-secure social media applications. Section 3 covers the experimental setup used, including acquisition types, forensic tools and methods used, and the chosen Android test platforms. Sections 4 and 5 reports the experimental results, and finally Section 6 concludes the paper.

2 Related Work

Messaging applications on mobile platforms arose from the low-cost of mobile internet compared to individual Short Message Service (SMS) messages [13]. Internet messaging also allows for the easy transmission of multimedia messages. Adoption of security measures such as encryption and ephemeral functions came from security concerns due to hacking attacks and privacy issues [14, 15]. Mutawa et al. [16] conducted research into three popular social networking applications: Facebook, Twitter and MySpace on iOS, Android and Blackberry platforms. The applications were installed on the test platforms and scenarios were created by logging in and using their features, inputting a list of keywords that would later be used in searching for artefacts. The acquisition of data was performed by rooting the device, which was necessary as without rooting, areas of the device where some artifacts are stored cannot be accessed [16]. Wu et al. [17] investigated two ephemeral messaging applications: SnapChat and Burner on iOS and Android platforms. Physical acquisition of the platforms and subsequent analysis revealed that the transient data from SnapChat, once expired, was not securely removed, and in some cases only the filename had been changed, which lead to recovery of artefacts. Since the release of SnapChat, ephemeral messaging applications have adopted a focus on security. The latest includes applications such as Wickr and Telegram, which function as a normal messaging application, but offer a heightened focus on security.

Walnycky et al. [18] conducted investigations into a wide range of mobile messaging applications. Using the recovery and analysis of the database file for each respective app, a group of applications were analysed, including Wickr, which was identified as having no vulnerabilities, as in no artefacts were recovered due to encryption. Another investigation into Wickr carried out by Mehrotra et al. [19] using a pre-analysis scenario creation method found no artefacts related to Wickr. The investigation employed a backup of the Android platform and used string searches to analyse the acquired data. The lack of results was caused by Wickr's extensive use of encryption in its data storage techniques.

Satrya et al. [20] analysed Telegram on Android devices, finding that the application uses a database to store messages, as well as other storage locations for sent and received files. The methodology used in [20] highlighted three areas of interest, including the Telegram ".apk" installer package, the internal storage directory, and the "cache4.db" database. The "cache4.db" database was SQLite formatted, and was logically acquired using a backup program before being analysed with SQLite Database Browser.

3 Experimental Setup

The investigation reported in this paper focussed on the ephemeral messaging functions on both Wickr and Telegram and also the channel feature of Telegram. As well as expanding on the research carried out by Satyra et al. [20] by incorporating Telegram's Channel, the experiments were conducted using physical data acquisition and analysis, which has not been previously carried out in the context of both applications. A variety of different forensic analysis methods were used to recover artefacts from both Wickr and Telegram. Due to the differences in the security measures taken by the applications, such as encryption, the methods and artefacts for each vary. The flow of experimental work started with the initial step of scenario creation, followed by logical analysis of data. If data examined were encrypted, then the methodology focussed on the analysis of the application itself and its ephemeral data. If plaintext data were recovered, analysis of the artefacts and ephemeral data followed. In any case, Random Access Memory (RAM) acquisition and analysis were completed to recover artefacts.

3.1 Data Acquisition

The acquisition methods used in this paper can be divided into two categories, logical and physical. Logical data acquisition in this context refers to the use of the device's filesystem to recover files. In this case, the chosen method of logical acquisition was ADB pull, a command from the Android Debug Bridge [21], a set of tools used for development and testing of Android applications.

The physical acquisition method used in this context was to create a bit-for-bit copy of the target platforms' "user" partition, which on Android stores all files related to user activity, including both application information and personal files. To create the copy, the tool Data Dump (dd) [22] was used. Other Android partitions, including the cache and the system partition were also acquired and analysed using a string search, however in this case no related artefacts were recovered, so these partitions were not included in the scope of the investigation.

In addition to the recovery of data from the Android platform's secondary storage, the Random Access Memory (RAM) of the device was also acquired. An application for android called Memory Dump [23] was used for this purpose. Memory Dump allows the user to choose an application running on the platform and acquire the memory associated with that application. This is useful in cases of encryption, as encrypted data may be stored in plaintext format in RAM when in use, for ease of access by the application.

3.2 Wickr

Wickr is a highly secure messaging application. Offering "privacy by design" [9], its simple easy to use interface hides an array of anti-forensics techniques, such as encryption of local data and network traffic [18], as well as ephemeral messaging features. In the face of such features, forensics techniques such as pre-analysis scenario creation and string searching acquired data fall down [18, 19]. Instead of recovering artefacts, the application and its effect on the Android platform, were analysed for a greater understanding in a forensic context.

The Wickr application itself is stored on the Android platform in the form of an ".apk" installer package. Inside this package is a file called "classes.dex", which is a file containing all the definitions for the functions of the application. To extract the "classes.dex" file, the ".apk" was opened using the archive extraction utility in Windows, and the file was extracted. To analyse the file, it was converted to a Java archive. A specialised tool exists for this purpose, called "dex2jar" [24]. The output from this tool is a Java archive that was analysed using another tool, Java Decompiler [25]. When analysed, the file revealed all the class definitions for Wickr, which gives an insight into how it operates.

To analyse the files associated with Wickr, the application's data directory ("/data/data/com.mywickr.wickr2") was logically acquired. To find out how the ephemeral data function works in Wickr, the data directory was acquired multiple times at different stages of the test scenario. These were: before messages were received, after messages had been received, after messages had been removed by the ephemeral function and after the "secure shredder" had been run. The "secure shredder" offers the shredding of deleted data. The various acquired versions of the directories were analysed and compared to look for any changes in files as well as the size of Wickr's database "wickr_db".

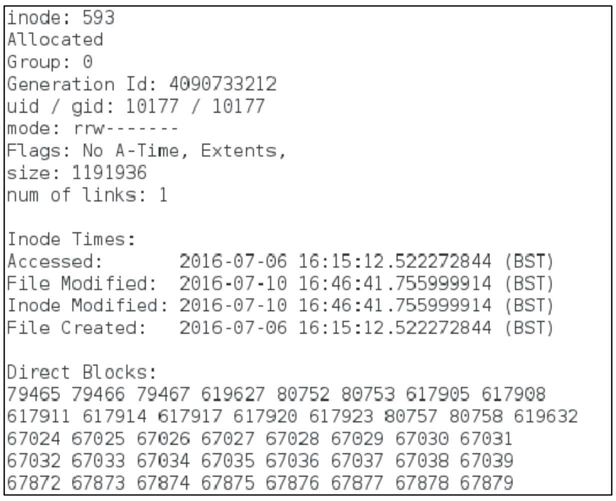
The physically acquired image was analysed in Autopsy 3.0.8 [26]. Autopsy is the graphical frontend for a set of Linux forensics tools called the Sleuthkit. This contains tools that allow for the recovery of deleted data. Autopsy also allows for the processing of unallocated space, which is an important part of the analysis as ephemeral messaging functions rely on the deletion of data. Artefacts such as files sent as attachments to messages that had been deleted via the ephemeral messaging function were recovered using Autopsy.

3.3 Telegram

Telegram is a feature-rich messaging application that also incorporates security into its operation, by adding end-to-end encryption as well as ephemeral messaging. Telegram offers a unique way of communication called the channel feature. Introduced in 2015 with the version 3.2.1 [27], this feature allows any user to start a channel which can either be public or private. Any other Telegram user can locate public channels using the search function, whereas private channel rely on an invite-only URL based system. Once the channel is established, owners and admins broadcast material to all subscribers. Telegram's channel feature was used by an active terrorist organisation to disseminate propaganda [28], which highlights the potential for the features use in major crime, and creates the need for forensic analysis.

The work carried out by Satrya et al. [20] established methods of recovering messages sent via Telegram, recovering artefacts by logically acquiring the "cache4.db" database file from the platform, which was analysed using SQLite Database Browser [29]. In our work, the channel feature was analysed using this methodology to reveal additional artefacts related to channels. In addition to this, artefacts related to the ephemeral messaging function were recovered by analysing the physically acquired image using Autopsy, as mentioned in section 3.2.

To locate where remnant channel messages may be stored on the test platform, the filesystem clusters of the "cache4.db" database had to be identified. As Android uses a UNIX-like filesystem, EXT4, this information is stored in a meta-structure known as an "inode". After identifying the "inode" of the "cache4.db" database by mounting the physically acquired image and using the "ls" command, the tool "istat" from the Sleuthkit [26] was used to identify filesystem clusters. . The tool "istat" was used to print out the "inode" statistics from the physically acquired image for examination of data. The output of "istat" can be seen in Figure 1 - under "Direct Blocks", importantly, the clusters on which the file was stored can be seen.



```
inode: 593
Allocated
Group: 0
Generation Id: 4090733212
uid / gid: 10177 / 10177
mode: rrw-------
Flags: No A-Time, Extents,
size: 1191936
num of links: 1

Inode Times:
Accessed:       2016-07-06 16:15:12.522272844 (BST)
File Modified:  2016-07-10 16:46:41.755999914 (BST)
Inode Modified: 2016-07-10 16:46:41.755999914 (BST)
File Created:   2016-07-06 16:15:12.522272844 (BST)

Direct Blocks:
79465 79466 79467 619627 80752 80753 617905 617908
617911 617914 617917 617920 617923 80757 80758 619632
67024 67025 67026 67027 67028 67029 67030 67031
67032 67033 67034 67035 67036 67037 67038 67039
67872 67873 67874 67875 67876 67877 67878 67879
```

Fig. 1. Output of the "istat" command for Telegram's "cache4.db" database.

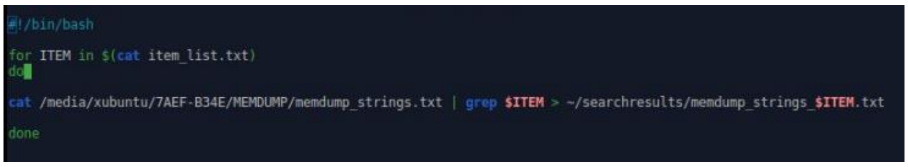
To examine the clusters, the physically acquired image was opened in the hex editor WinHex [30]. To locate the data from the list of clusters, a simple calculation [31] was made to find the byte offset of the clusters:

$$B = C * SpC * S \tag{1}$$

Where B is the byte offset in bytes, C is the cluster number, SpC is the sectors per cluster and S is the sector size in bytes. In the case of the test platform, the SpC was 8 and S was 512, so the calculation was performed as per the chosen cluster. A specific add-on for Telegram, offered by Oxygen Forensic Suite 2014 [32], was used to recover encrypted messages. Oxygen Forensic Suite 2014 also provides a file browser for viewing and recovering files stored as attachments, however, Oxygen Forensic Suite 2014 is limited to active files, i.e. files that have not been deleted [33]. Later versions (2015 onwards) do support recovery of deleted files [32]; however, in this case, Autopsy 3.0.8 was used.

3.4 Linux Utilities

When examining data on a Linux forensic workstation, basic file utilities, such as “ls”, “cat”, “strings” and “grep” were used to examine directory structures and files. The tool “strings” extracts all string format data from a file. A bash script was created to perform string searches on the output of the strings utility. The script is shown in Figure 2. The first part of the script’s command, cat, sends the saved “strings” output to the second part, grep, which searches for matches to a keyword dictionary, and saves the results to a new file. This technique was used to search the physically acquired images for keywords relating to the respective application, as well as the live memory acquisition files.



```
#!/bin/bash

for ITEM in $(cat item_list.txt)
do

cat /media/xubuntu/7AEF-B34E/MEMDUMP/memdump_strings.txt | grep $ITEM > ~/searchresults/memdump_strings_$ITEM.txt

done
```

Fig. 2. String search bash script.

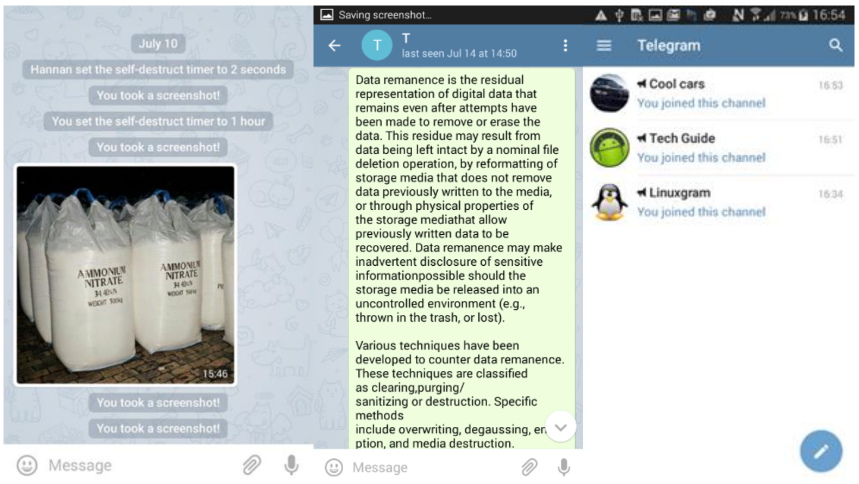
3.5 Platforms

A Samsung Galaxy S4 Mini [34] was the primary platform used for the experimental setup. An AllWinner A13 Android tablet [35] was used as an alternate platform in order to ensure the repeatability of all experiments performed, as well as sending and receiving messages to and from the test platform. The choice of test platform in this case, a phone from Samsung’s flagship galaxy range, reflects the current state of the worldwide smartphone market, which is dominated by Android [36], the market for which is in turn dominated by Samsung [36]. Another reason for choosing Android was its large online developer community, which stems from its open source status. The applications were both installed on the platform using Android’s built-in app store, Google Play [1]. The test platforms and the application versions used are listed in Table 1.

Table 1. Android platforms and application versions.

Name	Specifications				
	Model number	**Android version**	**Kernel version**	**Wickr version**	**Telegram version**
Samsung Galaxy S4 Mini [34]	GT-I9195I	4.4.4 (KitKat)	3.10.28-5334500	2.6.4.1	3.10.1
AllWinner A13 [35]	Q8	4.4.2 (KitKat)	3.4.39	2.6.4.1	N/A

Once the applications were installed on the test platform, scenarios were created by creating accounts on the respective applications and using the features, to simulate a suspect’s device. As the apps differ in their features somewhat, the artefacts creation procedure also slightly differed. To test both the regular and ephemeral messaging functions, messages were sent with identifiable text, as well as attachments, to the test platform. For Wickr, these messages simulated a conversation which included an attachment, a picture (.jpg) file, as well as some key words.


July 10

Hannan set the self-destruct timer to 2 seconds

You took a screenshot!

You set the self-destruct timer to 1 hour

You took a screenshot!

You took a screenshot!

You took a screenshot!

T
last seen Jul 14 at 14:50

Data remanence is the residual representation of digital data that remains even after attempts have been made to remove or erase the data. This residue may result from data being left intact by a nominal file deletion operation, by reformatting of storage media that does not remove data previously written to the media, or through physical properties of the storage mediathat allow previously written data to be recovered. Data remanence may make inadvertent disclosure of sensitive informationpossible should the storage media be released into an uncontrolled environment (e.g., thrown in the trash, or lost).

Various techniques have been developed to counter data remanence. These techniques are classified as clearing,purging/ sanitizing or destruction. Specific methods include overwriting, degaussing, en ption, and media destruction.

Cool cars — You joined this channel — 16:53

Tech Guide — You joined this channel — 16:51

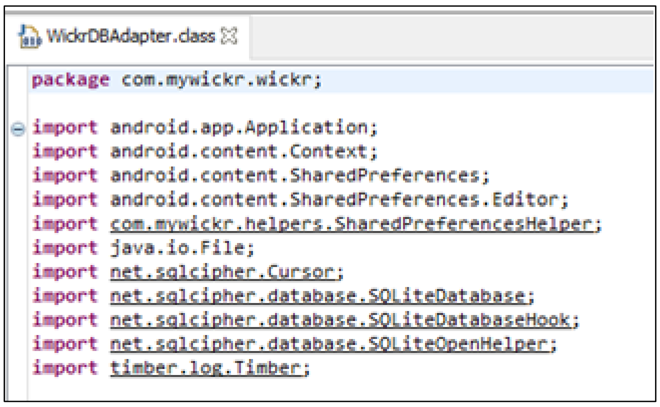
Linuxgram — You joined this channel — 16:34

Fig. 3. Established scenario on Telegram.

For Telegram, the scenario creation involved both the creation of, and subscription to channels. In addition to creating custom channels, a number of public channels were subscribed to also, which meant that the messages recieved via these channels were included in the list of potential artefacts. Unlike Wickr, in Telegram’s scenario more messages were required to utilise the channel function. For this purpose, bulk text was copied from the internet, as seen in Figure 3. The range of artefacts recovered from both standard conversations and channels included message text, files (images and documents) and usernames. Due to the inclusion of channel function in Telegram, the number of artefacts incoroprated in the analysis was more than Wickr's.

4 Results for Wickr

Analysis of the installer package for Wickr revealed the techniques it uses to store data. When applications store data in databases, they sometimes use a programming object called a "Database Helper". In Wickr's case, the class of functions called "WickrDBAdapter.class" reveals the use of "SqlCipher", a database encryption library for SQLite. The use of such a library explains why any messages recieved by Wickr were not stored in plain text. An extract from the "WickrDBAdapter.class" is shown in Figure 4.



```
package com.mywickr.wickr;

import android.app.Application;
import android.content.Context;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import com.mywickr.helpers.SharedPreferencesHelper;
import java.io.File;
import net.sqlcipher.Cursor;
import net.sqlcipher.database.SQLiteDatabase;
import net.sqlcipher.database.SQLiteDatabaseHook;
import net.sqlcipher.database.SQLiteOpenHelper;
import timber.log.Timber;
```

Fig. 4. Extract from "WickrDBAdapter.class".

Acquisition and analysis of Wickr's associated data as described in Section 3.2 revealed how the ephemeral function of Wickr worked. The results of comparison of the acquired data are shown in Table 2, which show that on a logical level, the presence of encrypted ".wic" files correlates with the status of stored messages in Wickr. The results in Table 2 reveal that the ephemeral function removed the data, at least logically. The process of the ephemeral function caused the start of file decay: firstly the files were logically deleted.

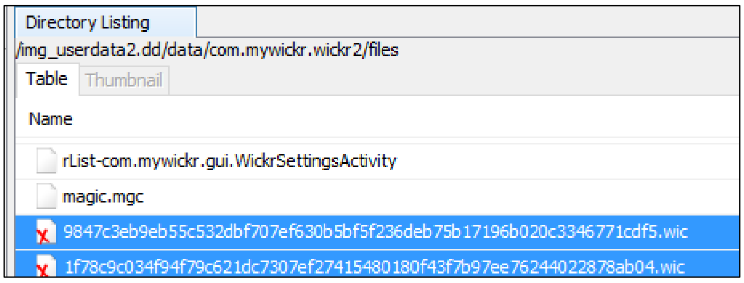
The proposed operation of Wickr's file decay is shown in Table 3, which demonstrates the varying levels of file storage and deletion in Wickr at the stages listed in Table 2. As the results of Table 2 rely on logical acquisition, analysis of the physically acquired images of the Android platform was required to further investigate the deletion process and recover deleted artefacts. Navigating to the Wickr data directory on the physically acquired image using Autopsy revealed filesystem references to previously present ".wic" files. A screenshot of this is shown in Figure 5.

Table 2. Wickr data analysis results.

Stage of file removal (arbitrary)	**Secure shredder status**	**Copy taken**	**Files directory and further observations**
1	Before	Before images were received	Only two ".wic" files, "pcc.wic" and "pcd.wic" were present in the files directory.
2	Before	After images were received	Two ".wic" files, each with 64 character string file names, were present in the files directory. Their sizes were 47488 and 54136 bytes respectively.
3	Before	After images were removed	Two ".wic" files were not present.
4	After	After images were removed	Two ".wic" files were not present.

Table 3. Model of Wickr's file decay.

Stage of file removal (arbitrary)	**1**	**2**	**3**	**4**
Status of received file	N/A	File present, encrypted, stored in .wic file	File present, encrypted, filesystem header removed	File overwritten with random or null data
Process required to recover file	N/A	Logical level acquisition, for example copy.	Low level acquisition, such as device data dump or chip-off analysis	File unrecoverable.



/img_userdata2.dd/data/com.mywickr.wickr2/files

Name
rList-com.mywickr.gui.WickrSettingsActivity
magic.mgc
9847c3eb9eb55c532dbf707ef630b5bf5f236deb75b17196b020c3346771cdf5.wic
1f78c9c034f94f79c621dc7307ef27415480180f43f7b97ee76244022878ab04.wic

Fig. 5. Files subdirectory of Wickr's data directory in Autopsy.

Finally, analysis of the RAM dump acquired using the method described in Section 3.1 revealed that, much like the other areas of storage, Wickr does not expose much data and analysis found very few artefacts. Using the string search method described in Section 3.4, the dumped RAM data was analysed. The results are shown in Table 4.

Any field within the "< … >" parenthesis represents information which was omitted for privacy purposes.

Table 4. Results of Wickr's RAM dump string search.

Search expression	Reason	Results
wickr	Name of the App.	Returned paths of files in the data directory of Wickr, as well as extracts from various core Wickr libraries.
<Hidden>	Password used to register wickr account	No Matches.
cccumobilef	Username used to register wickr account	Matching String found in *dumped__7428b000-7428e000_rw-p* dump file.
ocelotorus	Username of account used in scenario	No matches.
an.jpg	Filename of picture uploaded – file extension added because "an" too ambiguous	No matches
invicta	See above	No matches.
goods	Excerpts from messages sent in pre-analysis scenario creation	No pertinent matches.
Yes	See above.	No pertinent matches.
meet	See above.	No pertinent matches.

The results in Table 4 show that Wickr does not store information such as received messages in plaintext in RAM. However, a single artefact, the name of the account used to sign up to Wickr, was found. The account name could be used in co-operation with Wickr and telecom services to locate the user that signed up using the captured device. This artefact was not reported in the work carried out previously by Walnycky et al. [18] and Mehrotra et al. [19], which used searching of Wickr's stored data.

5 Results for Telegram

The standard messaging capabilities of Telegram are divided into two separate features: normal and secret chats. Normal chats use a server to store messages and employ client-server encryption, while during secret chats the involved platforms communicate with each other directly and employ end-to-end encryption [10]. Upon analysis using SQLite Database Browser, the messages received via channels were stored in the same way as normal messages, established before by Satyra et al. [20], with an additional artefact, the link to join a private channel. In the database, these artefacts was identified using the UID column, which was unique to the conversation. Figure 6 shows the suspect channel, "test channel (create)", identified in the chats table of the "cache4.db" database. The UID was captured from the selected entry as shown in Figure 6. Records were retrieved from the messages table using this UID. The status of the out column in the messages table identifies the suspect's platform as being an owner or admin of the suspect channel, as the value of "1" in the out column shown in Figure 6 denotes the messages were sent. Corresponding entries in the chat settings table revealed the private URL which was used to join the channel, as seen in Figure 7.

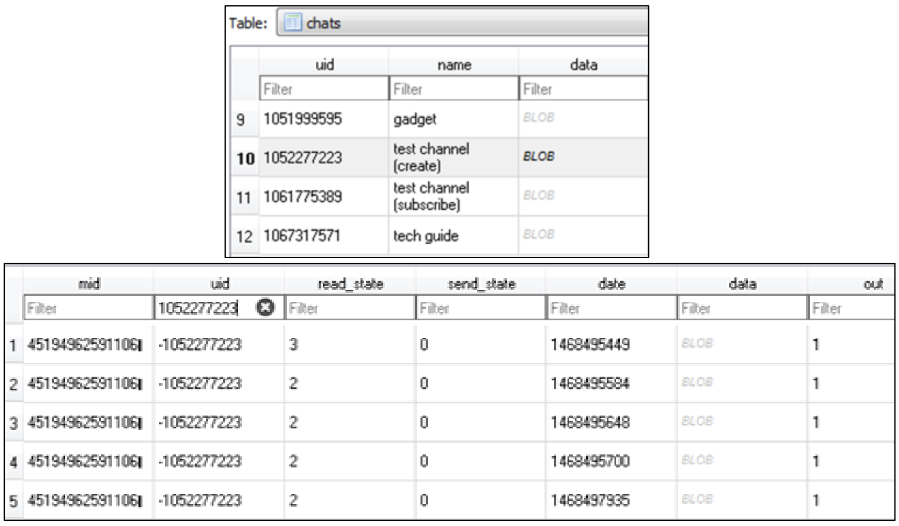
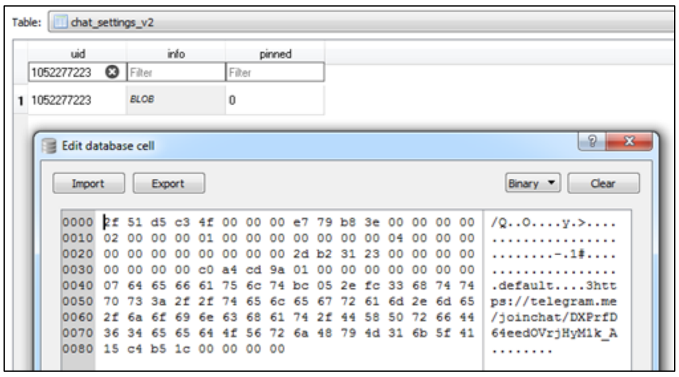


Table: chats

	uid	name	data
	Filter	Filter	Filter
9	1051999595	gadget	BLOB
10	1052277223	test channel (create)	BLOB
11	1061775389	test channel (subscribe)	BLOB
12	1067317571	tech guide	BLOB

	mid	uid	read_state	send_state	date	data	out
	Filter	1052277223	Filter	Filter	Filter	Filter	Filter
1	45194962591106	-1052277223	3	0	1468495449	BLOB	1
2	45194962591106	-1052277223	2	0	1468495584	BLOB	1
3	45194962591106	-1052277223	2	0	1468495648	BLOB	1
4	45194962591106	-1052277223	2	0	1468495700	BLOB	1
5	45194962591106	-1052277223	2	0	1468497935	BLOB	1

Fig. 6. Suspect channel in Chats Table of Telegram's "cache4.db" database.

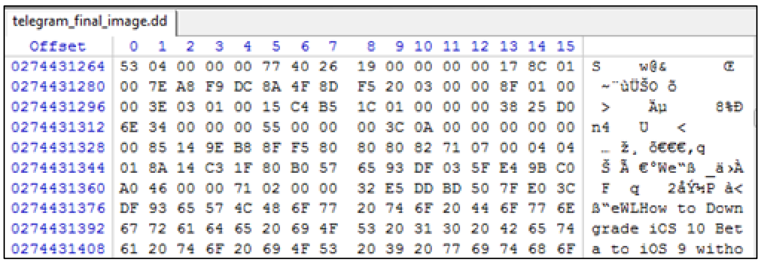


	uid	info	pinned
	1052277223	Filter	Filter
1	1052277223	BLOB	0

```
0000 2f 51 d5 c3 4f 00 00 00 e7 79 b8 3e 00 00 00 00  /Q..O....y.>....
0010 02 00 00 00 01 00 00 00 00 00 00 00 04 00 00 00  ................
0020 00 00 00 00 00 00 00 00 2d b2 31 23 00 00 00 00  ........-.1#....
0030 00 00 00 00 c0 a4 cd 9a 01 00 00 00 00 00 00 00  ................
0040 07 64 65 66 61 75 6c 74 bc 05 2e fc 33 68 74 74  .default....3htt
0050 70 73 3a 2f 2f 74 65 6c 65 67 72 61 6d 2e 6d 65  ps://telegram.me
0060 2f 6a 6f 69 6e 63 68 61 74 2f 44 58 50 72 66 44  /joinchat/DXPrfD
0070 36 34 65 65 64 4f 56 72 6a 48 79 4d 31 6b 5f 41  64eedOVrjHyM1k_A
0080 15 c4 b5 1c 00 00 00 00                          ........
```

Fig. 7. Retrieval of channel join link in Telegram's "cache4.db" database.

In a real world scenario, it is possible that channels are un-followed or other anti-forensics measures are taken preceding the capture of the target platform. In order to examine messages left over from un-followed or deleted channels, the "cache4.db" file was examined on the imaged physical user partition. Using WinHex to examine the data at the byte with the technique mentioned in Section 3.3 revealed artefacts from the public Telegram channels subscribed to during the scenario creation. These messages remained in the “cache4.db” database even though the channel had been unfollowed and was no longer accessible via the normal interface. Figure 8 shows a message viewed in WinHex at byte offset 274431264. Files received via channels were stored in the “Telegram” directory on the internal storage, which has the absolute path of “/data/media/0/Telegram/” [20]. Upon unfollowing of channels, the files received from these channels are still present, lacking signs of deletion.



```
telegram_final_image.dd
  Offset      0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
0274431264   53 04 00 00 00 77 40 26  19 00 00 00 00 17 8C 01   S    w@&      Œ
0274431280   00 7E A8 F9 DC 8A 4F 8D  F5 20 03 00 00 8F 01 00    ~¨ùÜŠO õ
0274431296   00 3E 03 01 00 15 C4 B5  1C 01 00 00 00 38 25 D0    >    Äµ     8%Ð
0274431312   6E 34 00 00 00 55 00 00  00 3C 0A 00 00 00 00 00   n4   U   <
0274431328   00 85 14 9E B8 8F F5 80  80 80 82 71 07 00 04 04    … ž¸ õ€€€‚q
0274431344   01 8A 14 C3 1F 80 B0 57  65 93 DF 03 5F E4 9B C0    Š Ã €°We"ß _ä›À
0274431360   A0 46 00 00 71 02 00 00  32 E5 DD BD 50 7F E0 3C    F  q   2åÝ½P à<
0274431376   DF 93 65 57 4C 48 6F 77  20 74 6F 20 44 6F 77 6E   ß"eWLHow to Down
0274431392   67 72 61 64 65 20 69 4F  53 20 31 30 20 42 65 74   grade iOS 10 Bet
0274431408   61 20 74 6F 20 69 4F 53  20 39 20 77 69 74 68 6F   a to iOS 9 witho
```

Fig. 8. Messages from unfollowed channel.

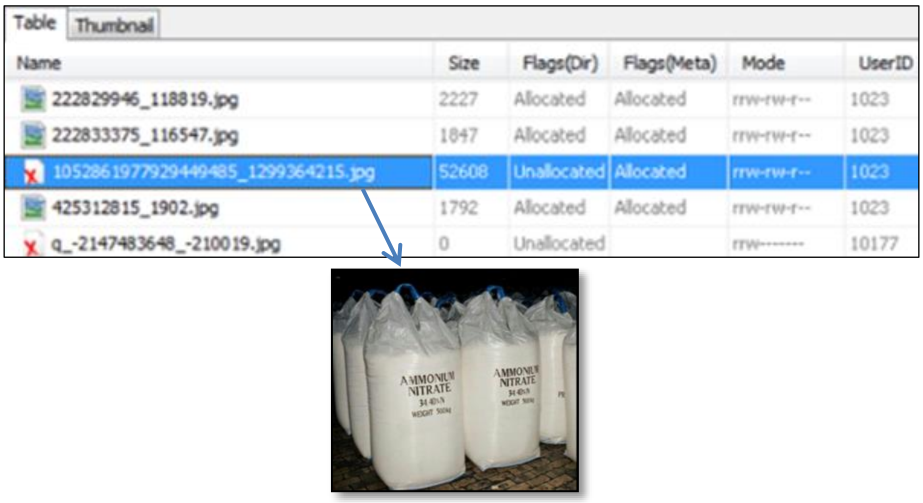


Table | Thumbnail

Name	Size	Flags(Dir)	Flags(Meta)	Mode	UserID
222829946_118819.jpg	2227	Allocated	Allocated	rrw-rw-r--	1023
222833375_116547.jpg	1847	Allocated	Allocated	rrw-rw-r--	1023
1052861977929449485_1299364215.jpg	52608	Unallocated	Allocated	rrw-rw-r--	1023
425312815_1902.jpg	1792	Allocated	Allocated	rrw-rw-r--	1023
q_-2147483648_-210019.jpg	0	Unallocated		rrw-------	10177

Fig. 9. Recovery of expired image using Autopsy.

The secret chat in Telegram offers an ephemeral messaging function, the self-destruct timer, which allows users to set expiry dates for the conversation. Before the message from the test scenario expired, examination of Telegram’s cache directory located at “/data/media/0/Android/data/org.telegram.messenger/cache” [17] showed the presence of the file, stored as “1052861977929449485_1299364215.jpg”, as well as few accompanying thumbnails. Upon expiry, the message was no longer accessible from the normal interface, and was no longer present in the cache. However, this expired image file was recovered in the cache directory on the physical image when Autopsy’s file browser tool was used, as shown in Figure 9. It was found that in this case, the file system entry was removed after the expiry; however, the file was still present in the device. These findings reveal that Telegram uses a logical deletion process to handle ephemeral message attachments, allowing for the recovery of expired multimedia messages using physical acquisition and examination of the captured device.

Table 5. Results of Telegram's RAM analysis.

Search Expression	**Reason**	**Results**
<First name>	First name used to create Telegram account, potentially identifying artefact.	Matching string found in dumped__78ea2000-78fd7000_rw-p and dumped__7a993000-7a99c000_rw-p dump files.
<Second Name>	Surname name used to create Telegram account, potentially identifying artefact.	Matching string found in dumped__78ea2000-78fd7000_rw-p and dumped__7a993000-7a99c000_rw-p dump files.
<Phone Number>	Phone number used to sign up to Telegram account, potentially identifying artefact.	No matches found.
+44	National mobile telephony prefix	A number of matches related to Graphical User Interface (GUI), but no identifying artefacts.
magnetic	Excerpts from messages sent in pre-analysis scenario creation	A number of matching strings found in dumped__77cb0000-77cb9000_rw-p dump file, among others.
overwrite	See above	A number of matching strings found in dumped__77cb0000-77cb9000_rw-p dump file, among others.

Table 5 reports the results of Telegram's RAM analysis which was acquired using the method described in Section 3.1 and analysed using the string search method described in Section 3.4. Results revealed the recovery of many plaintext artefacts. Fields within the "< … >" parenthesis represents confidential information which was omitted due to privacy purposes.

6 Conclusions

The experiments in this paper provided understanding of the manner in which secure messaging applications like Wickr and Telegrams store their data. Results revealed that the Wickr stored it's recieved messages in encrpyted ".wic" files in the data directory. While the messages did cause the production of ".wic" files, they did not affect the size of the application's database. Unlike Wickr, in the case of Telegram, the standard forensics tools had built-in modules to recover encrypted artefacts. Our analyses documented these artefacts in geater detail, including information relating to the joining of channels, and the recovery of messages after the channels had been unfollowed. The RAM dump technique for Wickr recovered a plaintext account username, which was valid only in the context of live analysis where the phone had not been turned off since being captured. Like Wickr, The RAM dump was also succesful in recovering Telegram's plaintext artifacts, which shows the integrity of this method in analysing similar applications. The direction of future research will focus on the decryption techniques and also the application of proposed methods in analysing similar SMAs on other popular platforms such as iOS and Windows Phones.

References

1. Google Play, https://play.google.com/store?hl=en_GB, [Accessed: September 2016]
2. Almasy, S., Meilhan P., Bittermann, J.: Paris Massacre: At least 128 killed in gunfire and blasts, French officials say, http://edition.cnn.com/2015/11/13/world/paris-shooting/, (2015), [Accessed: September 2016]
3. Madi, M., Ryder, S., Macfarlane, J., Beach, A., Park, V.:As it happened: Charlie Hebdo attack, http://www.bbc.co.uk/news/live/world-europe-30710777, (2016), [Accessed: September 2016]
4. Roussinous, A.: The social media Accounts of British Jihadis in Syria just got a lot more distressing, http://www.vice.com/en_uk/read/british-jihadis-beheading-prisoners-syria-isis-terrorism, (2014), [Accessed: September 2016]
5. Torok, R., http://theconversation.com/how-social-media-was-key-to-islamic-states-attacks-on-paris-50743, (2015), [Accessed: 20th July 2016]
6. Vidino, L., Hughes, S.: ISIS in America: From retweets to Raqqa, http://www.stratcomcoe.org/download/file/fid/2828, (2015), [Accessed: September 2016]
7. Apple App Store, http://www.apple.com/uk/itunes/, [Accessed: September 2016]
8. Perklin, M., https://www.defcon.org/images/defcon-20/dc-20-presentations/Perklin/DEFCON-20-Perklin-AntiForensics.pdf, (2012)
9. Wickr Official Website, https://www.wickr.com, [Accessed: September 2016].
10. Telegram Official Website, https://telegram.org, [Accessed: September 2016].
11. Anderson, M.:The demographics of device ownership, http://www.pewinternet.org/2015/10/29/the-demographics-of-device-ownership/, (2015), [Accessed: September 2016]
12. SnapChat, http:// mwpartners.com/snapchat-is-now-the-third-most-popular-social-network-among-millennials/, (2014), [Accessed: September 2016]
13. Barot, T., Oren, E.: Guide to Chat Apps, http://towcenter.org/research/guide-to-chat-apps/, (2015), [Accessed: September 2016]
14. Amir, W.: Viber to Put Full End-to-End Encryption on Their Messaging App, https://www.hackread.com/viber-end-to-end-encryption-on-messaging-app/ (2016), [Accessed: September 2016]
15. Mathur, N.: Facebook Messenger joins WhatApp in end-to-end encryption, http://www.livemint.com/Consumer/llIJ9Est0ZZIYfmvRSsTZP/Facebook-Messenger-joins-WhatsApp-in-endtoend-encryption.html , (2016), [Accessed: September 2016]

16. Mutawa, N. A., Baggili, I., Marrington, A.: Forensic Analysis of Social Networking Applications on Mobile Devices. Digital Investigation, vol. 9, pp. 24-33, (2012)
17. Wu, C., Vance, C., Boggs, R., Fenger, T.: Forensic Analysis of Data Transience Applications in iOS and Android, http://www.marshall.edu/forensics/files/Wu-Poster. pdf, (2013), [Accessed: September 2016]
18. Walnycky, D., Baggili, I.,Marrington, A., Moore, J., Breitinger, F. :Network and device forensic analysis of Android social-messaging applications. Digital Investigation, vol. 14, pp. 77-84, (2015)
19. Mehrotra, T., Mehtre, B. M.:Forensic analysis of Wickr application on android devices. IEEE International Conference on Computing Intelligence and Computing Research, pp. 1-6, (2013)
20. Satrya, G. B., Daely, P. T., Nugroho, M. A.: Digital Forensic Analysis of Telegram Messenger on Android Devices. 10th International Conference on Information and Communication Technology and System, Indonesia, (2016)
21. ADB tool, https://developer.android.com/studio/command-line/adb.html, [Accessed: September 2016]
22. Linux Man Page, http://linux.die.net/man/1/dd, [Accessed: September 2016]
23. Memory Dump, https://play.google.com/store/apps/details?id=com.cert.memdump&hl=en, [Accessed: September 2016]
24. Dex2Jar tool, https://github.com/pxb1988/dex2jar, [Accessed: September 2016]
25. Java Decompiler tool, http://jd.benow.ca, [Accessed: September 2016]
26. SleuthKit tool, http://www.sleuthkit.org, [Accessed: September 2016]
27. Telegram Channel, https://telegram.org/blog/channels, (2015), [Accessed: September 2016]
28. Cuthbertson, A., http://www.ibtimes.co.uk/isis-telegram-channel-doubles-followers-9000-less-1-week-1523665, (2015), [Accessed: September 2016]
29. DB Browser for SQLite Official Website, http://sqlitebroswer.org, [Accessed: September 2016]
30. X-Ways Forensics: WinHex, https://www.x-ways.net/winhex/index-m.html, [Accessed: September 2016]
31. Sedory, D. B.: Drive Offset and Sector Conversions, http://thestarman.pcministry.com/asm/mbr/DriveOffsets.htm, (2012), [Accessed: September 2016]
32. Oxygen Forensics Official Website, http://www.oxygen-forensic.com, [Accessed: September 2016]
33. Shortall, A., Azhar, M. A. H. B.: Forensic acquisitions of WhatsApp data on popular mobile platforms. Sixth International Conference on Emerging Security Technologies (EST), IEEE Press, Technische Universitaet Braunschweig, Germany, pp.13-17, (2015)
34. Samsung Galaxy Mini Official Web Page, http://www.samsung.com/uk/consumer/mobile-devices/smartphones/galaxy-s/GT-I9195ZKABTU, [Accessed: September 2016]
35. Allwinner A13 User Manual, http://linux-sunxi.org/A13, [Accessed: September 2016]
36. Woods, V., Meulen, R. V. D.: Gartner Says Worldwide Smartphone Sales Grew 3.9 Percent in First Quarter of 2016, http://www.gartner.com/newsroom/id/3323017, (2016), [Accessed: September 2016]