# Forensic Analysis of LINE Messenger on Android

Ming Sang Chang<sup>1\*</sup>, Chih Yen Chang<sup>2</sup>



<sup>1</sup> Department of Information Management, Central Police University, Taiwan mschang@mail.cpu.edu.tw

<sup>2</sup> Graduate Institute of Communication Engineering, National Taiwan University, Taiwan gsmmcc@gmail.com

Received 14 August 2016; Revised 07 February 2017; Accepted 13 February 2017

Abstract. Instant messaging has changed the way people communicate with each other. It is used by a wide range of age groups and backgrounds. Instant Messaging applications like LINE, Whatsapp, and WeChat which facilitate users to send and receive chat messages, video, audio and images via various end devices in real time. Its extensive use in everyday life provides unique opportunities but means that it can also be used to commit crime such as cyber stalking, cyber bullying or by becoming a medium for criminals' communication. In order to identify crimes, it is essentially required to retrieve these traces and evidences by using appropriate forensic technique. This paper studies the artifacts left by LINE application with Android and presents evidence gathering of Line messenger application. It proves beneficial for forensic analysts and practitioners as it assists them in course of mapping and locating digital evidences of Line messenger on Android.

Keywords: digital forensics, instant messaging, investigation, LINE

# 1 Introduction

Instant messaging (IM) is a type of online chat program which offers real-time text as well as audio, video, and image files transmission over the Internet. IM allows effective and efficient communication, allowing immediate receipt of acknowledgment or reply. Instant messenger applications such as LINE, WhatsApp, WeChat, and Facebook Messenger are some of the most widely used applications. The smart phones, tablet computers, personal computers, and the convenience of Internet made the use of such applications very popular.

LINE is a proprietary instant messenger application on smartphones, tablet computers and personal computers. LINE users can exchange texts, images, video and audio. They also can conduct free VoIP conversations and video conferences. LINE first launched in Japan in 2011 [1]. It reaches 100 million users within eighteen months and 200 million users only six months later [2]. LINE became Japan's largest social network in 2013. In October 2014 LINE announced that it had attracted 560 million users worldwide with 170 million active user accounts [3-4]. In the second quarter of 2016 LINE announced more than 220 million monthly active users [5].

LINE is a cross platform application with versions available for Windows, MAC, iOS, Android, Windows Phone, etc. It is a widely used and universal application. As the use of LINE is increasing rapidly, it is important to take measures in advance from forensic standpoint forecasting the potential use of it in cybercrimes such as hacking, copyright infringement, cyber stalking, and cyber bullying. To solve IM based cybercrimes, investigator need to perform forensic analysis of suspect device to find digital evidences.

User devices and IM applications may hold the data that can provide evidence of the activities carried out through them. The use environment of the IM applications can provide evidences. These evidences

<sup>\*</sup> Corresponding Author

can be used to profile the behavior of its user and may even allow the investigator to anticipate the users' actions [6-8]. Each device and application has its own acquisition requirements and potential sets of evidence.

It challenges to forensic examiners for recovering digital evidences of a conversation under investigation. In this work, we study and report the forensic analysis of an instant messaging namely LINE on Android system. In this paper, we implement forensic analysis procedures on the LINE messenger. It proves beneficial for forensic analysts and practitioners as it assists them in course of mapping and locating digital evidences of Line messenger on Android. This paper has organized as follows. In section 2 introduces the background and related works. In section 3, we outline the research methodology. In section 4, results and analysis are discussed. In section 5, we discuss our research findings. Finally, section 6 is a conclusion.

# 2 Related Works

The evidences were stored on three principle areas by using IM. They are hard drive, memory, and network. Some IM services have the ability to log information on the user's hard drive [9]. To use an IM, an account must be established to create a screen name provided with user information. Some instant messenger providers might assist the investigation with information of the account owner.

Evidence can be found in various internet file caches used by Internet Explorer for volatile IM and each cache holds different pieces of data. Apart from the normal files, files left by instant messenger on a hard drive can be in temp file format and will generally be deleted could be very difficult to retrieve once the machine is power down. An operating system generally stores information of all the installed and uninstalled applications in the system. The uninstalled application also leaves evidence. If a user has deleted an instant messenger application, there is a chance that a record can be found in the registry to prove that the instant messenger has once installed onto the system. Information is also stored within the memory. Since every application requires memory to execute, it is logical to think that there evidence could be left behind in the system's memory. The analysis on live memory has allows us to extend the possibility in providing additional contextual information for any cases. For any Windows based operating system, it is important evidence can usually be found beneath the physical memory, hibernation file and pagefile [10].

Artifacts of instant messaging have been of interest in many different digital forensic studies. Early work focused on artifacts left behind by many instant messaging applications, such as MSN Messenger [11], Yahoo Messenger [12], and AOL Instant Messenger [13]. In 2013 Mahajan et al. [14] performed forensic analysis of Whatsapp and Viber on five android phones using UFED and manual analysis. Cosimo Anglano [15] carried out Whatsapp forensics on Android in 2014 using YouWave virtualization platform. Levendoski et al. [16] concluded that artifacts of the Yahoo Messenger client produced a different directory structure on Windows Vista and 7. Wong et al. [17] and Al Mutawa et al. [18] demonstrated that artifacts of the Facebook web-application could be recovered from memory dumps and web browsing cache.

Said et al. [19] investigated Facebook and other IM applications, it was determined that only BlackBerry Bold 9700 and iPhone 3G/3GS provided evidence of Facebook unencrypted. Sgaras et al. [20] analyzed Skype and several other VoIP applications for iOS and Android platforms. It was concluded that the Android apps store far less artifacts than of the iOS apps. Chu et al. [21] focused on live data acquisition from personal computer and was able to identify distinct strings that will assist forensic practitioners with reconstruction of the previous Facebook sessions. Iqbal et al. [22] studied the artifacts left by the ChatON IM application. The analysis was conducted on an iPhone running iOS6 and a Samsung Galaxy Note running Android 4.1. Walnycky et al. [23] added that artifacts of the Facebook Messenger could vary depending on user settings, OS version, and manufacturer. Azfar et al. [24] adapt a widely used adversary model from the cryptographic literature to formally capture a forensic investigator's capabilities during the collection and analysis of evidentiary materials from mobile devices.

To our knowledge, no detailed analysis of LINE artifacts on Android devices has been undertaken, hence this research aims to fill the gap and provide a road map of LINE forensic artifacts. In this work, we implement forensic analysis procedures on the LINE messenger. We study the artifacts left by LINE application with Android and presents evidence gathering of Line messenger application.

## 3 Methodology

In our research, we use virtual machines with a standard installation of Windows 10 OS. The BlueStacks application was installed on Windows 10. Then we root the BlueStacks. The BlueStacks App Player is designed to enable Android applications to run on Windows PCs and Macintosh computers. BlueStacks emulates the Android OS within its own environment. The LINE application V5.8.1 was installed on the BlueStacks. We set up 15 different configurations and analyze them. We don't re-configure and copy volatile memory and non-volatile memory. This allowed us to examine a variety of test in several configurations and to facilitate forensic analysis of LINE Messenger. When conducting analysis with message exchange of using LINE Messenger, one of the main issues is to identify where potential data remnants resides. We focus on identifying data remnants of the activities of LINE on Android. This is suspected. Our research also includes the circumstances of using anti-forensic methodology to hide evidence, and whether remnants remain to identify the use of LINE Messenger.

This research focuses on what data remnants on Android after a user log in, exchange message, and keeps files of the use of LINE Messenger. We want to find username, password, text, and files. In addition, we also create circumstances to simulate a user running *CCleaner* to remove evidences. There are 15 virtual machines which replicate different circumstance of activities to gather the data in relation to the use of LINE on Android. We make multiple scenarios to explore the use of LINE. The virtual machines will be created for each different circumstance of LINE activities. This represents different Android phones available for analysis, with different circumstances and data remnants available for analysis on each VM. According to the activities of LINE, we create a base VM or base image file and 14 different VMs. The virtual machines reduce the costs of the study, since neither many real Android phones are necessary to carry out the experiments.

The base image file is to compare the subsequent image files to determine the changes made. It is possible to observe the changes of file systems. Our experimental test-bed consists of a set of virtual machines. That is VMware Workstation V12.0.0. For each experiment, the BlueStacks was installed on every virtual machine. The LINE Messenger V5.8.1 was installed on all virtual machines. In each experiment, we assign a role to each virtual device. We use it to carry out the corresponding activities. At the end of the experiment, we suspend the virtual device. We parse the file implementing the corresponding volatile memory and non-volatile memory by means of WinHex 17.9.0.0 and EnCase V7.04. Then we extract the files where LINE Messenger stores the data it generates. We also use Root Explorer V3.3.7 to analyze the LINE database files.

According to the activities of LINE, we create seven sub-experiment systems. They are Base-VM, Login-VM, Snd-VM, Rcv-VM, Keep-VM, Delete-VM, and Delete\_Keep-VM. In all experiments, there are 15 virtual machines to gather the data in relation to the activities of LINE as shown in Table 1.

| Virtual Machine | LINE activity   |
|-----------------|---|
| Base-VM         | No action   |
| Login-VM        | Login (Email address)                                 |
|                 | Send text message                                     |
| Snd-VM          | Send image file                                       |
|                 | Send video file                                       |
|                 | Receive text message                                  |
| Day VM          | Receive image file                                    |
| KCV-VIVI        | Receive video file                                    |
|                 | Receive Word file                                     |
| Koon VM         | Keep text message                                     |
| Keep-v IVI      | Keep Image file                                       |
| Delete VM       | Delete all data (sending and receiving)               |
| Delete- v M     | Delete all data with CCleaner (sending and receiving) |
| Dalata Kaan VM  | Delete all keeping data                               |
|                 | Delete all keeping data with CCleaner                 |

Table 1. All virtual machines with LINE activity

The different actions undertaken are as follows. We divide them in seven cases.

- The first case was to install the LINE messenger into base virtual machine with Android.
- The second case was to make a copy of the base virtual machine. An account of LINE was created for these experiments. We use email address to sign in LINE. We do nothing and sign out. Then we use Root Explorer V3.3.7 to analyze LINE database files and use WinHex 17.9.0.0 and EnCase V7.04 to analyze the data remnants to find the account and password.
- The third case was to make three copies of the base virtual machine for each scenario. There are three scenarios for different actions such as sending text, image, and video. After sending action we sign out and analyze LINE database files, volatile memory, and non-volatile memory to find the data remnants.
- The forth case was to make four copies of the base virtual machine for each scenario. There are four scenarios for different actions such as receiving text, image, video, and word files. After receiving action we sign out and analyze LINE database files, volatile memory, and non-volatile memory to find the data remnants.
- The fifth case was to make two copies of the base virtual machine for each scenario. There are two scenarios for different actions such as keeping text, and image. We use the keep feature of LINE to save text message, and image in cloud storage. After keeping action we sign out and analyze LINE database files, volatile memory, and non-volatile memory to find the data remnants.
- The sixth case was to make two copies of the base virtual machine for each scenario. We do the same actions as case 3 and 4. Then we delete all the sending and receiving text, image, video, and files. We log out and analyze LINE database files, volatile memory, and non-volatile memory to find the data remnants. In the other scenario, after we delete all the sending and receiving text, image, video, and files; we use CCleaner to remove LINE and delete temporary, history, cookies, recycle bin, memory dumps, log files, etc. Then we analyze LINE database files, volatile memory, and non-volatile memory to find the data remnants.
- 7. The seventh case was to make two copies of the base virtual machine for each scenario. We do the same actions as case 5. Then we delete all the keeping text, and image. We log out and analyze LINE database files, volatile memory, and non-volatile memory to find the data remnants. In the other scenario, after we delete all the keeping text, and image; we use CCleaner to remove LINE and delete temporary, history, cookies, recycle bin, memory dumps, log files, etc. Then we analyze LINE database files, volatile memory, and non-volatile memory to find the data remnants.

# 4 Result and Analysis

## 4.1 Login-VM

**Login with email address.** We analyze VMEM file and find the email address (testabc2016@gmail.com) but can't find the password (justtest2016) as shown in Fig. 1. In VMDK file the remnant of password can't be found but email address was found on C\ProgramData\BlueStacks\Android\Data.sparsefs\Store. The email address was also found in LINE database file *naver\_line* on /data/data/jp.naver.line.android/ database as Fig. 2. We believe the password with a secure encryption method on LINE Messenger V5.8.1 so that we can't find it. In this experiment, a search for the login password produced no matches in the forensic image and memory dump.

| 65A4DFC0 |                                       |                              |
|----------|---------------------------------------|------------------------------|
| 65A4E000 | g https://android.clients.c           | google.ccm/fdfe/details?doc= |
| 65A4E040 | jp.naver.line.android/account=testabo | 2016@gmail.com x¬a Q         |
| 65A4E080 | µ[n7Q uÿj Q                           | Accept-Ranges none           |
| 65A4E0C0 | Cache-Control private,                | max-age=600 Content          |
| 65A4E100 | -Type application/x-gzip              | Date Tue, 17 Nov 2           |
| 65A4E140 | 015 02:59:07 GMT Expires              | Tue, 17 Nov 2015 02:59:07    |
| 65A4E180 | GMT Server GSE                        | Transfer-Encoding ch         |

Fig. 1. The remnants of email address on memory

| NOTI_TIMELINE_NOTIFICATION | true                  |
|----------------------------|-----------------------|
| IDENTITY_PROVIDER          | 1                     |
| IDENTITY_IDENTIFIER        | testabc2016@gmail.com |
| PROFILE_ACCOUNT_MIGRATION  | true                  |
| SUGGEST_ACCOUNT_SETTING    | true                  |
| SUGGEST_REGISTER_PHONE     | true                  |

Fig. 2. The remnants of email address on naver\_line

#### 4.2 Snd-VM

**Sending text message.** In volatile memory the sending user's name (Yaaichu) can't be found, but receiver's name (HsinHsin), chat message (Send you a message), time stamp (1447732785173), and the receiver ID (u300bc27d6c3e5fe3390861c4ae9f240d) are shown in Fig. 3. In VMDK files the remnant of receiver's name (HsinHsin) can only be found. The text message was also found on chat\_history table in naver line database file as Fig. 4.

| 3EFEDF00 | 84f872514d3c40b794c4afc24ef795a31024abc7  Q ″U: P€íÙ# G O             | Γ |
|----------|---|---|
| 3EFEDF40 | ) ik u300bc27d6c3e5fe3390861c4ae9f240dHsinHsinH                       |   |
| 3EFEDF80 | sinHsinGo!Go!Go! ó% ``0m0472bfba725175649326211913f63d1b3d92fdc73     |   |
| 3EFEDFC0 | 37f/0m0472bfba725175649326211913f63d1b3d92fdc7337f ÿ Q ″U: P€ù+>      |   |
| 3EFEE000 |   |   |
| 13A34680 | 3477833580448 <mark>u300bc27d6c3e5te3390861c4ae9t240dmarijuana</mark> | 1 |
| 13A346C0 | 102714477328500581447732855122 u ' 0 3'' 347782                       |   |
| 13A34700 | 9932768u300bc27d6c3e5fe3390861c4ae9f240dSend you a message 14477      |   |
| 13A34740 | 327851731447732793102 v 0 5'' 3477828068503u3                         |   |

Fig. 3. The remnants of sending text message on memory

| u085311ecd9e3e3 | 3d74ae4c9f5437cbcb5 | u085311ecd9e3e3d74ae4c9f5437cbcb5 | 【重要】其他裝置的LINE要求使用您   |
|-----------------|---------------------|-----------------------------------|----------------------|
| u085311ecd9e3e3 | 3d74ae4c9f5437cbcb5 | u085311ecd9e3e3d74ae4c9f5437cbcb5 | LINE系統提醒您,您目前已登入LINE |
| u085311ecd9e3e3 | 3d74ae4c9f5437cbcb5 | u085311ecd9e3e3d74ae4c9f5437cbcb5 | 提醒您:您的LINE帳號已經在其他裝   |
| u085311ecd9e3e  | 3d74ae4c9f5437cbcb5 | u085311ecd9e3e3d74ae4c9f5437cbcb5 | 【重要】 其他裝置的LINE要求使用您  |
| u0o0bc27d6c3e5  | fe3390861c4ae9t2484 |                                   | I test a experiment. |
| u300bc27d6c3e5  | fe3390861c4ae9f240c |                                   | Send ycu a message.  |
| u320bc27d6c3e5  | fe3390861c4ae9f2460 |                                   | marijuana1027        |
| created time    | delivered time      |                                   |                      |
| 1447729545291   | 0                   |                                   |                      |
| 1447729582512   | 0                   |                                   |                      |
| 1447732185236   | 0                   |                                   |                      |
| 1447732380531   | 0                   |                                   |                      |

Fig. 4. The remnants of text on chat history table

**Sending image.** The image file is found on memory as Fig. 5. In VMDK files the locations of remnants of image file are found as Table 2. The remnants of image file are also found on chat\_history table in *naver line* database file as Fig. 6.

Table 2. The locations of remnants of image file

```
C\ProgramData\BlueStacks\Logs\BlueStacksUsers.log
C\ProgramData\BlueStacks\UserData\SharedFolder
C\$Extend\$UsnJrnl·$J
C\$MFT
C\ProgramData\BlueStacks\Android\Data.sparsefs\Store
```

| 3B23D040 | 28906000~/.0~/.8~/.0.0~/./storage/sdcard/windows/BstSharedFolder   |
|----------|--|
| 3B23D080 | /IMG 1028.JPG -1360557281  1  BstSharedFolder content://media/exte |
| 3B23D0C0 | rnal/images/media/8~/.1238628906000~/.0~/.8~/.0.0~/./storage/sdd   |
| 3B23D100 | ard/windows/BstSharedFolder/IMG 1028.JPG                           |
| 3B23D140 |  |
| 67D7BD00 |  |
| 67D7BD40 | 0 )' { u300bc27d6c3e5fe3390861c4ae9f240d9246                       |
| 67D7BD80 | 44772000001448939311866 s "file:///mnt/sdcard/windows/BstSharedF   |

Fig. 5. The remnants of sending image file on memory

| u085311ecd9e3e3d7     | 4ae4c9f5437cbcb5    | u0853     | 11ecd9e3e3d7   | 4ae4c9f5437  | cbcb5 | 提醒您:您的LIN         |
|-----------------------|---------------------|-----------|----------------|--------------|-------|-------------------|
| u085311ecd9e3e3d7     | 4ae4c9f5437cbcb5    | u0853     | 11ecd9e3e3d7   | 4ae4c9f5437  | cbcb5 | 提醒您:您的LIN         |
| u085311ecd9e3e3d7     | 4ae4c9f5437cbcb5    | u0853     | 11ecd9e3e3d7   | '4ae4c9f5437 | cbcb5 | 【重要】其他裝           |
| u085311ecd9e3e3d7     | 4ae4c9f5437cbcb5    | u0853     | 11ecd9e3e3d7   | 4ae4c9f5437  | cbcb5 | LINE系統提醒您         |
| u085311ecd9e3e3d7     | 4ae4c9f5437cbcb5    | u0853     | 11ecd9e3e3d7   | 4ae4c9f5437  | cbcb5 | 提醒您:您的LIN         |
| u085311ecd9e3e3d74ae4 | c9f5437cbcb5 u08531 | lecd9e3e3 | 3d74ae4c9f5437 | cbcb5 提醒您:   | 您的LIN | E¢                |
| 0300bc27d6c3e5fe33    | 390861c4ae9f240a    | >         |                |              |       |                   |
| attachement_image     | attachement_image   | height    | attachement    | _image_width | atta  | chement_local_uri |
| 0                     |                     |           |                |              | T L   |                   |
| 0                     |                     |           |                |              | † L   |                   |
| 0                     |                     |           |                |              | T 📖   |                   |
| 0                     |                     |           |                |              | I     |                   |

Fig. 6. The remnants of image on chat history table

Sending video. The remnants of sending video file are the same as Sending image.

In these three experiments the remnants can be found in volatile memory and non-volatile memory. When a user sends a file using the LINE app, there will be records remaining in BlueStacks application and Windows system files such as \$MFT, \$UsnJrnl-\$J to indicate the filenames, and directory paths for the sending files. The remnants are also recorded in memory as Fig. 3 & 5.

#### 4.3 Rcv-VM

**Receiving text message.** In volatile memory the sending user's name (Yaaichu) can't be found, but receiver's name (HsinHsin), chat message (Receive your messages.), time stamp (1448959859549), and the receiver ID (u300bc27d6c3e5fe3390861c4ae9f240d) are shown in Fig. 7. The remnant of receiver's name (HsinHsin) can only be found on C\ProgramData\BlueStacks\Android\Data.sparsefs \Store in VMDK files. The text message is also found on chat history table in *naver line* database file.

| 6E2BAE80 | GO)ik                    | u300bc27d6c3e5fe33                    | 90861c4ae9f24 |
|----------|--------------------------|---------------------------------------|---------------|
| 6E2BAEC0 | OdHsinHsinHsinHsinGo!Go! | Go! ó¾"`0m0472bfba725175649           | 326211913£63d |
| 6E2BAF00 | 1b3d92fdc7337f/0m0472bfb | a725175649326211913f63d1b3d           | 92fdc7337f ÿ  |
| 6E2BAF40 | Q\»Èl P€ù+> 1 O          | ik u085311                            | ecd9e3e3d74ae |
| 6E2BAF80 | 4c9f5437cbcb5LINELINE0m0 | 52184f872514d3c40b794c4afc2           | 4ef795a31024a |
| 6E2BAFC0 | bc7/0m052184f872514d3c40 | b794c4afc24ef795a31024abc7            | Q\»úÚ P€íÙ#   |
| 622D9980 | <u> </u>                 | 3543023503503 <mark>u300bc27d6</mark> | c3e5fe3390861 |
| 622D99C0 | c4ae9f240d1300bc27d6c3e5 | fe3390861c4ae9f240dtt drugs           | 1448959875467 |
| 622D9A00 | 0EMTVER 4 ' OO9'         | 3543022378063u                        | 300bc27d6c3e5 |
| 622D9A40 | fe3390861c4ae9f240du300b | c27d6c3e5fe3390861c4ae9f240           | dReceive your |
| 622D9A80 | messages. 14489598595490 | EMTVER 4 00 '                         | 3543          |

Fig. 7. The remnants of receiving text on memory

**Receiving image.** The image file is found on memory as Fig. 8. The location of remnants of image file is found on C\ProgramData\BlueStacks\Android\Data.sparsefs\Store in non-volatile memory. The remnants of image file are also found on chat history table in *naver line* database file

| 06926100 |  |
|----------|--|
| 06926140 | 61k !'/ #  |
| 06926180 | <pre>/storage/sdcard/Pictures/LINE/1448978058</pre>            |
| 069261C0 | 125.jpg *8 V]¦ŠV]¦Šimage/jpeg14489780581251448978058125.jpg Q] |
| 06926200 | Ò< ``àoy« y<-1876327261LINE X 1 G                              |
| 06926240 | /storage/sdcard/Pictures/LINE0 V]¦Š R 1 =                      |
| 06926280 | /storage/sdcard/Pictur   |
| 069262C0 | es0 V]¦‱ W 1 G /s  |

Fig. 8. The remnants of receiving image file on memory

**Receiving video & Receiving word file.** The remnants of receiving video file and word file are the same as **Receiving image.** 

In these four experiments the remnants can be found in volatile memory and non-volatile memory. When a user receives a text message or a file using the LINE app, there will be records remaining in BlueStacks application, Windows system files and LINE database files to indicate the contents, filenames, and directory paths for the downloaded files.

## 4.4 Keep-VM

**Keep text message.** The remnants can't be found in VMDK files. In volatile memory a keeping message (test\_test\_550) is shown in Fig. 9. The *contentItems* database table on *com\_linecorp\_linebox\_ android* database file in /data/data/jp.naver.line.android/database was found. The text message, file size, and time stamp can be found in *contentItems* database table as Fig. 10.

| 6D065F00 | } ''           | )       | ad_kp  | 9227fb  | 38a8f  | 73b6ff41 | c6f135 | 59800 | 93c 648e |
|----------|----------------|---------|--------|---------|--------|----------|--------|-------|----------|
| 6D065F40 | 8e64d5694a4186 | 2d927de | 442cf3 | Bf test | _test  | 550test  | _test_ | 550{  | "urlList |
| 6D065F80 | ":[]}{}w       | )       | a      | ad_kp 9 | 227fb3 | 38a8f73b | 6ff41c | 6f13  | 5980093c |

Fig. 9. The remnants of keeping text message on memory

| clien | tld          |                 |     |                  |                  |       |      |
|-------|--------------|-----------------|-----|------------------|------------------|-------|------|
| ad_k  | p 9227fb38a  | 8f73b6ff41c6f13 | 5   | 980093c eb71c4af | 367245bba5042472 | 292ab | 09ae |
| ad_k  | p 9227fb38a  | 8f73b6ff41c6f13 | 5   | 980093c 648e8e64 | d5694a41862d927  | de442 | cf3f |
| ad_k  | p 9227fb38a  | 8f73b6ff41c6f13 | 5   | 980093c 506cbe90 | f5dd495090a859cb | 5c90l | ofd4 |
| size  | title        | text            | ī   | createdTime      | modifiedTime     | total | Size |
| 5     | agric37      | agric37         | Ī   | 1449036951292    | 14490369512921   | 7     |      |
| 13    | test test 55 | test test 550   | † I | 1449037231525    | 1449037231525    | 13    |      |
| 8/    | bulats51     | bulats51        | t   | 1449037307843    | 1449037307843    | 8     |      |

Fig. 10. The remnants of keeping text on *contentItems* table

**Keep image.** In non-volatile memory the locations of remnants of image file are shown in Table 3. In volatile memory the remnants of image file are shown in Fig. 11. The image file, file size, directory, and time stamp can be found in *contentItems* database table. The remnants can also be found on *contents* and *sourceInfo* database table.

Table 3. The locations of remnants of image file

C\ProgramData\BlueStacks\Logs\BlueStacksUsers.log C\\$LogFile C\ProgramData\BlueStacks\Android\Data.sparsefs\Store C\\$Extend\\$UsnJrnl•\$J C\ProgramData\BlueStacks\UserData\SharedFolder C\\$MFT

| L | 2970EE00 | f\ % GqA % ad_kp 81  |
|---|----------|--|
| L | 2970EE40 | 4f04eed73011d142ab0ca8a0329a0a 070a996c50984b09acfb0ae742d9c197              |
| L | 2970EE80 | <sup>v</sup> UIMG_1225.jpg file:///storage/sdcard/Android/data/ip.naver.line |
| L | 2970EEC0 | .android/cache/lineKeep/tuna/29/IMG 1225.jpgEile:///storage/sdca             |
| L | 2970EF00 | rd/Android/data/jp.naver.line.android/cache/lineKeep/mackerel/98             |
| L | 2970EF40 | d45aadee355a83d5f0c0914e3ff211{"IMAGE UPLOAD OPTION EXTRA KEY":1             |
| L | 2970EF80 | , "originalTime": 1332835682000, "height": 960, "width": 1280} linekeep      |
| L | 2970EFC0 | b2d3275715964f13a77dc7b94b8c9b06_881e23t039ac557pIMG_1225.jpg{}              |

Fig. 11. The remnants of keeping image on memory

#### 4.5 Delete-VM

**Delete all data.** We do all the sending and receiving actions as above experiments. Then we delete all of sending and receiving data. The locations of remnants are shown in Table 4. The contents of *contacts* and *chat\_history* database table have been deleted and can't find remnants. The file name can be found in memory dump except the text message.

Table 4. The locations of remnants of deleting all data

| C\\$Extend\\$UsnJrnl·\$J                               |
|--|
| C\ProgramData\BlueStacks\Android\Data.sparsefs\Store   |
| C\\$LogFile  |
| C\\$MFT  |
| C\ProgramData\BlueStacks\Android\SDCard.sparsefs\Store |

**Delete all data with CCleaner.** We do the same actions as **Delete all data** and use CCleaner to remove LINE apps and delete temporary, history, cookies, recycle bin, memory dumps, log files, etc. The locations of remnants are only on C\\$Extend\\$UsnJrnl.\$J. The directory jp.naver.line.android has been deleted and any remnants can't be found in database files. In memory dump the file name can be found except the text message.

#### 4.6 Delete\_Keep-VM

**Delete all keeping data.** We do all the keep actions as **Keep-VM**. Then we delete all of the keeping data. The locations of remnants are shown in Table 5. The chat message can't be found in database file. The file name can be found in memory dump except the text message.

**Delete all keeping data with CCleaner.** We do the same actions as **Delete all keeping data** and use CCleaner to remove LINE apps and delete temporary, history, cookies, recycle bin, memory dumps, log

files, etc. The locations of remnants are only on C\\$Extend\\$UsnJrnl\\$J. The directory *jp.naver*. *line.android* has been deleted and any remnants can't be found in database files. The file name can be found in memory dump except the text message.

| Table 5. The | locations | of remnants | of deleting | all keeping | data |
|--------------|-----------|-------------|-------------|-------------|------|
|--------------|-----------|-------------|-------------|-------------|------|

| C\\$Extend\\$UsnJrnl·\$J                             |  |
|--|--|
| C\ProgramData\BlueStacks\Android\Data.sparsefs\Store |  |
| C\ProgramData\BlueStacks\Logs\BlueStacksUsers.log    |  |
| C\\$LogFile  |  |
| C\\$MFT  |  |
| C\ProgramData\BlueStacks\UserData\SharedFolder       |  |
| C\Lost Files\HD-LogRotatorService.exe                |  |
| C\Lost Files\HD-Adb.exe                              |  |

#### 5 Discussions

In this research, we identified artifacts for LINE application. We focus on both the volatile memory and non-volatile memory artifacts. Our experiments showed that the LINE apps on volatile memory has proved that critical application data is present in the RAM and it can be extracted for further analysis. Our non-volatile memory analysis has shown that LINE application activities remain some artifacts in different locations. This indicated that when a user has used the LINE apps, there will be records remaining in the application folder. While opening the LINE database files we know the LINE database schema and find the remnants. The critical application data is present in the database tables.

It should be noted that the significance and location of artifacts could be investigated. In our research, it was determined that: (1) LINE apps maintain directories in the application folders. (2) LINE apps hold the database schema for the application caches. (3) The LINE apps caches copies of the transferred and downloaded files in the application folder.

Our examinations of the physical memory and database table captures indicated that the memory dumps and database table can recover the application caches in plain text, with the exception of the login password. The fact that there was no clear text password in the volatile memory and non-volatile memory should perhaps be encrypted. We performed all our research inside a virtual machine which gave us an advantage to download or run executable files without having to worry about any executable affecting the host machine. Other than that all our forensic data was not leaked to the outside world and a separate environment was provided to hold all our files in one place. We saw that data is stored in various forms by the developer and each location can become a treasure for an investigator. The artifacts findings are summary in Table 6.

| Virtual Machine |                         | Volatile Memory | Non-Volatile Memory | Root Explorer |
|-----------------|-------------------------|-----------------|---------------------|---------------|
| Login-VM        | Email addr              | Found           | Found               | Found         |
| Snd-VM          | Text                    | Content found   | Found               | Content found |
|                 | Image file              | Found           | Found               | Found         |
|                 | Video file              | Found           | Found               | Found         |
| Rcv-VM          | Text                    | Content found   | Found               | Content found |
|                 | Image file              | Found           | Found               | Found         |
|                 | Video file              | Found           | Found               | Found         |
|                 | Word file               | Found           | Found               | Found         |
| Keep-VM         | Text                    | Content found   | Found               | Content found |
|                 | Image file              | Found           | Found               | Found         |
| Delete-VM       | Delete                  | Found           | Found               | Not found     |
|                 | Delete with<br>CCleaner | Found           | Found               | Not found     |
| Delete_Keep-VM  | Delete                  | Found           | Found               | Not found     |
|                 | Delete with<br>CCleaner | Found           | Found               | Not found     |

 Table 6. Summary of findings

## 6 Conclusions

Instant messaging is increasingly popular among individuals and business organizations. Applications such as LINE, WhatsApp, WeChat, and Facebook Messenger are some of the commonly used applications. With the tremendous use of such applications, it may be used to commit crimes. It is important to identify the forensic artifacts left by these application. In this paper we have presented the findings from our forensic examination of LINE instant messaging application with Android. The study consists of installation, uninstallation, logins, conversations, transferred files, and other LINE activities. The results indicated that use of the LINE for Android leave useful evidential material on the volatile memory and non-volatile memory.

In this paper, we study and report the forensic analysis of an instant messaging namely LINE on Android system. Because the limitation of experiment cost we use BlueStacks to emulate the Android OS system. The implementation may vary between different end devices. In the future, we can use the real different smart phone with Android to get the artifacts left by LINE application. The different operating OS with LINE application can also be studied such as iOS and Windows 10.

# References

- [1] About LINE Corporation, ABOUT. <a href="http://linecorp.com/en/company/info">http://linecorp.com/en/company/info</a>, 2016 (accessed 16.08.08).
- [2] E. Lukman, LINE hits 200 million users, adding 100 million in just 6 months. <a href="https://www.techinasia.com/line-hits-200-million-users-adding-100-million-users-6-months">https://www.techinasia.com/line-hits-200-million-users-adding-100-million-users-6-months</a>>, 2016 (accessed 16.07.08).
- [3] H. Josh, LINE finally reveals it has 170 million monthly active users. <a href="https://www.techinasia.com/line-japanese-messaging-app-has-170-million-monthly-active-users">https://www.techinasia.com/line-japanese-messaging-app-has-170-million-monthly-active-users</a>, 2016 (accessed 16.07.10).
- [4] A. Akimoto, Looking at 2013's Japanese social-media scene. <a href="http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#.V6rgbtR97RY>">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#.V6rgbtR97RY>">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#.V6rgbtR97RY>">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#.V6rgbtR97RY>">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#.V6rgbtR97RY>">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#.V6rgbtR97RY>">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013s-japanese-social-media-scene-3/#">http://www.japantimes.co.jp/life/2013/12/17/digital/looking-at-2013/s\_japanese-social-media-scene-3/#">http://www.japanese-social-media-scene-3/#"</a>
- [5] LINE: number of monthly active users 2014-2016. <a href="http://www.statista.com/statistics/327292/number-of-monthly-active-line-app-users/">http://www.statista.com/statistics/327292/number-of-monthly-active-line-app-users/</a>>, 2016 (accessed 16.08.01).
- [6] A. Orebaugh, J. Allnutt, Data mining instant messaging communications to perform author identification for cybercrime investigations, in: S. Goel (Ed.), Digital Forensics and Cyber Crime, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, 2010, pp. 99-110.
- [7] A. Iqbal, H. Al Obaidli, A. Marrington, A. Jones, Windows surface RT tablet forensics, Digital Investigation 11(2014) S87-S93.
- [8] The United Nations Office on Drugs and Crime, Comprehensive study on cybercrime. <a href="https://www.unodc.org/documents/organized-crime/UNODC">https://www.unodc.org/documents/organized-crime/UNODC</a> CCPCJ EG.4 2013/CYBERCRIME STUDY 210213.pdf>, 2016 (accessed 16.08.06).
- [9] A.R. Gonzales, R.B. Schofield, D.W. Hagy, Investigations involving the internet and computer networks, National Institute of Justice, Washington, DC, 2007.
- [10] Y. Gao, T. Cao, Memory forensics for QQ from a live system, Journal of Computers 5(4)(2010) 541-548.
- [11] M. Dickson, An examination into MSN Messenger 7.5 contact identification, Digital Investigation 3(2)(2006) 79-83.
- [12] M. Dickson, An examination into Yahoo Messenger 7.0 contact identification, Digital Investigation 3(3)(2006) 159-165.
- [13] J. Reust, Case study: AOL instant messenger trace evidence, Digital Investigation 3(4)(2006) 238-243.
- [14] A. Mahajan, M.S. Dahiya, H.P. Sanghvi, Forensic analysis of instant messenger applications on android devices, International Journal of Computer Applications 68(8)(2013) 38-44.
- [15] C. Anglano, Forensic analysis of WhatsApp Messenger on Android smartphones, Digital Investigation 11(2014) 201-213.

- [16] M. Levendoski, T. Datar, M. Rogers, Yahoo! Messenger Forensics on Windows Vista and Windows 7, Digital Forensics and Cyber Crime, vol. 88, Springer Berlin Heidelberg, Berlin, Germany, 2012.
- [17] K. Wong, A.C.T. Lai, J.C.K. Yeung, W.L. Lee, P.H. Chan, Facebook forensics. <a href="https://www.fbiic.gov/public/2011/jul/facebook">https://www.fbiic.gov/public/2011/jul/facebook</a> forensics-finalized.pdf>, 2011 (accessed 16.08.01).
- [18] N. Al Mutawa, I. Al Awadhi, I. Baggili, A. Marrington, Forensic artifacts of Facebook's instant messaging service, in: Proc. International Conference for Internet Technology and Secured Transactions (ICITST), 2011.
- [19] H. Said, A. Yousif, H. Humaid, IPhone forensics techniques and crime investigation, in Proc: International Conference and Workshop on Current Trends in Information Technology, 2011.
- [20] C. Sgaras, M.-T. Kechadi, N.-A. Le-Khac, Forensics Acquisition and Analysis of Instant Messaging and VoIP Applications, Computational Forensics, vol. 895, Springer International Publishing, Cham, Switzerland, 2015.
- [21] H.-C. Chu, D.-J. Deng, J.H. Park, Live data mining concerning social networking forensics based on a Facebook session through aggregation of social data, IEEE Journal on Selected Areas in Communications 29(7)(2011) 1368-1376.
- [22] A. Iqbal, A. Marrington, I. Baggili, Forensic artifacts of the ChatON instant messaging application, in: Proc. 2013 Eighth International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE), 2013.
- [23] D. Walnycky, I. Baggili, A. Marrington, J. Moore, F. Breitinger, Network and device forensic analysis of Android socialmessaging applications, Digital Investigation 14(Supplement 1)(2015) S77-84.
- [24] A. Azfar, K-K.R. Choo, L. Liu, An android social App forensics adversary model, in: Proc. Annual Hawaii International Conference on System Sciences (HICSS 2016), 2016.