PUBLIC SERVICE FOR SMART CITY THROUGH INTERNET MESSENGER: WHICH MESSENGER PERFORM BETTER IN TERMS OF 'ANTI IDENTITY FRAUD'?

[1] KRESNA RIDWAN, [2] HATMA SURYOTRISONGKO, [3] ARIS TJAHYANTO, [4] BEKTI CAHYO HIDAYANTO

[1,2,3,4] Department of Information Systems, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

E-mail: [1] ridwankresna@gmail.com , [2] hatma@is.its.ac.id , [3] aristj@its.ac.id , [4] bekticahyo@is.its.ac.id

ABSTRACT

To deliver services to the public, the government needs to communicate with the resident. The easiest way to communicate without boundary is by using social media. This concept brings messenger applications such as Line, Whatsapp, and Telegram as a communication media for a smart city. However, of those three applications, which one is the safest to use? There are various threats such as cybercrime and identity fraud which make the government must choose the most appropriate application that can be applied as public service media. In this study the authors analyze that three application (Line, WhatsApp, and Telegram) in the most popular operating system, Android and iOS to assess which messenger application that has non repudiation characteristic – it means we should know the origin of data about who send the message to prevent an identity fraud in the development of e-government system in smart city. This research analyzed messenger storage location to get the desired data and perform further analysis into forensics analysis software. From this research WhatsApp messenger on every OS platform has the best performance. WhatsApp not only identifies user's id but also their phone number like Telegram Messenger. Also, WhatsApp has no encryption on their conversation data that make the examiner easier to solve a case on WhatsApp than Telegram. WhatsApp also manages their media data well, all media have names registered in the database, have an extension and neatly organized in the media folder.

Keywords: *Identity Fraud, Internet Messenger, Smart city*

1. INTRODUCTION

Messenger application is the most popular communication media used by people today because it is practical and easy to use. The most widely used messengers in Indonesia are Line, Whatsapp, and Telegram because they have additional security features and easy-to-understand even for aged people [1]. However, every application has advantages and disadvantages. Moreover, as time goes by, messenger applications today are widely used as a means to commit digital crimes. Some of the cases that occur are online fraudulent scams or identity fraud on behalf of a confident person not related to the case [2].

Apart from that, a messenger application requires an operating system so that it can be useful for a human. Operating system manages all of the smartphone resources and turns it into a service that users can use [3]. For mobile devices or smartphones, the most widely used operating systems are Android, iOS, and Windows phone. This fact can be ascertained from the statistical analysis of StatCounter, Android and iOS companies having the highest market share of other operating systems [4].

Security concept begins to be applied to mobile device application development. Currently, there are "appWatchdogs," http://viaforensics.com/appwatchdog/ which is a company-made viaForensics application that is in the field of digital forensics. This application is used to assess other mobile apps in managing their data. Mobile app criteria can be said to be safe according to appWatchdog are as follows:

a. How does the app manage web history and data cache?
b. Does the app safely deliver login data?
c. Can the application avoid MITM attacks?
d. Does the app safely transmit sensitive data?
e. Can the app be protected from hacking?

f. Can the app permanently delete its data and prevent the storage of unused data on the hardware?
g. Is the application able to handle interruptions safely?
h. Can the application backup data correctly?

From the above criteria the authors divide the need for application into three types as follows:

a. Examiner – to solve digital forensics criminal case
b. Casual user – for daily use
c. Crime – used for digital crime

The details of the criteria described in Table 1.

Table 1 Application criteria for each user

Examiner :
1. Can be analyzed by examiner/investigators 2. Has proper management of history and cache data 3. Store the login data properly 4. The application can save backup data properly
Casual User :
1. Can be analyzed by examiner/investigators 2. Store the login data properly 3. Has proper management of history and cache data 4. The application can delete unnecessary data permanently 5. The application can save backup data properly
Crime :
1. Application can delete unnecessary data permanently

From 3 types of usability above authors conclude that the most suitable messenger application for smart city development must be useful for both examiner and casual user. The application criteria based on three types above as follows:

a. Can store data properly
b. Can be analyzed by examiner/investigators
c. Store login data properly
d. The application can manage the history of data and cache properly

Softwares used in this research to perform forensics analysis are "Belkasoft Evidence Center" and "FTK Imager." This research performed based on experiment scenarios and dialogue made before that will be explained in the methodology.

2. RELATED WORKS

There is much research about mobile forensics because there are many mobile phone developer and criminal case behind them. The most critical researchers need to know about smartphone forensics are a crucial source of evidence needed by the investigator, such as the history of incoming and outgoing calls, the message transmitted, emails, browsing information, multimedia files, etc. Researchers have developed many ways to investigate smartphones such as investigating storage of smartphone, Live forensics, and Internet Protocol Forensics. Joe Sylve et al. [5], examined whether authentication credentials in the volatile memory of Android mobile devices can be discovered using open source forensics tools named Linux memory extractor (LiME) software. The application used to investigate were M-Banking app, e-shopping, password managers and encryption & data hiding application. The analysis result revealed that majority of the Android applications are vulnerable to the recovery of authentication credentials from the volatile memory. The authentication data stored at memory dump and only deleted if the smartphone restarted or the battery removed. The conclusion of this research that application developers should use correct and secure programming techniques and guidelines (i.e., delete the authentication credentials when they not used from applications), to avoid authentication fraud and enhance the level of privacy.

Another research about live forensics conducted by V thing[6]. In this research, authors perform live memory forensics analysis for mobile phone by investigating the behavior of the mobile phone's volatile memory. Authors using memory acquisition tool (memgrab) to perform memory acquisition. This research revealed there is differences length message between outgoing and incoming message, but still have high rates that can be used as evidence.

Now we move to windows mobile LiveSD Forensics by Eyup S. Canlar et al. In [7] authors proposed LiveSD Forensics to online device by acquiring data from RAM and EEPROM. Authors using Handheld Reverse Engineering Tool (HaRET) to replace the running windows mobile OS by a Linux kernel and Dump RAM to perform live data acquisition of the

RAM. This research revealed LiveSD Forensics is the only forensic tool, among the compared solution that can perform live forensics acquisition of both RAM and EEPROM of windows device mobile.

From the example above about Live forensics above, now we present the example of research about storage forensics. In [8] author examined telegram messenger on windows phone. Their methodology divides by three as follows: Open Knowledge, Analysis of artifacts and Learn the source code. The result of research is Open source instant messenger such as telegram on open source operating systems such as Android are more comfortable to analyze than closed source instant messenger such as WhatsApp on closed source operating systems such as iOS and windows phone. This causing, not every generated artifact can be examined by the examiner.

Normaziah et al. in [9], perform an analysis of extracting and analyzing data from an android based smartphone. Authors using Sleuth Kit Autopsy to perform analysis through devices. This work aims to discover method extracting and analyze data from an Android-based smartphone. This research revealed that an analysis software used determined how much artifacts generated. Another research named Cosmo Anglano do the same research about telegram aimed at Android devices. In [10], the authors divided methodology to three parts as follows: completeness (identification generated data by the application); repeatability (possibility of the third party to replicate the experiments under the same operational conditions to obtain the same result); generality (the result should represent all Android devices version). Authors using SQLite viewer to read database telegram and learn from telegram documentation about the data structure location. This research revealed how table data connect each other to provide data and information for user and examiner.

Time the message and file delivered recorded as a database in the timestamp table. There is research about how timestamp on messenger works by [11]. The research revealed that the timezone database version present in the Android distribution are lagging behind the currently released version of the database. Furthermore, need tools for normalization timestamp based on the Time Zone Database present on the device. Timestamp stored in database formatted as Unix Epoch Time [12]. We can convert UET format to readable format for human with a site such as https://www.epochconverter.com/. Not just Android devices, there are so many research about another platform such as iOS [13]. Husain and Sridhar perform research on messenger applications such as AIM, Yahoo! Messenger and Google Talk. Authors using logical acquisition data by backup data iTunes to get the physical image. This research revealed there are so many boundaries to do forensics on an iPhone because of its uniqueness and closed source. However, authors can acquire and restore data artifacts from the iPhone included username and password.

Some of the researchers do an investigation to KIK messenger stated on a journal by Olawale Surajudeen Adebayo et al. [14]. Authors used the same methodology as digital forensics methodology with a little modification for an existed case as follows: Android devices preparation, logical acquisition (need rooted device), and analyze data from Android devices. Research conducted shows that KIK messenger on the android artifact and data structure a little bit different than on iOS devices. This research can be used for investigator or examiner to interpreting criminal case that using KIK messenger as a media. Another researcher performs an investigation to KIK messenger on iOS devices [15]. The methodology used by authors divided by three steps as follows: The first step is preparation to jailbreaking device and installs software package manager 'Cydia.' Moreover, then Ovens do an installation of KIK Messenger and do the experiment. Moreover, the final step manually analyzes artifact data generated by KIK messenger using SQLite viewer and python script. Research conducted by authors could be used to investigate the actor who sends messages, sent content, structure location data. In the end, this research can be used as a best practice for the investigator to solve a criminal case about KIK messenger.

Another iForensics research from Christos Sgaras [16]. This research examines Whatsapp, Skype and Tango Messenger. Authors using Cellebrite UFED as a forensics analysis software to read artifact data generated those messenger application. There are three extraction methods used by authors as follows: logical extraction, file system extraction, and manual extraction. The research revealed a practical method to get artifact data from IM-application whether it is iOS or Android devices, even though there is some problem about possibilities and limitations of using session cloning from messenger and limitation of retrieving information from IM vendors.

Some of the messenger application provides a feature to secure their data called as 'private chat'. Research about private chat has conducted by G.B. Satrya et al. [17]. Investigation methodology used is same as the investigation process from digital forensics investigation framework (DFIF) but with a little modification to suit with a criminal case about private chat. The methodology used as follows: Identification, Preservation, Analysis, and Presentation. Authors using six scenarios to test how private chat works, and messenger applications used are Telegram, Line, and KakaoTalk. This research provides a guide for examiner to know the exact location of secret chat location to deal with cybercrime cases related to these three social messenger application. There is another research from G.B Satrya [18] about Digital Forensics Study of Line Artifact on Android OS. Authors explain about a description how to get artifacts generated by Line messenger. These Android forensics processes were divided into two processes as follows: Online forensics using live log forensics where the smartphones kept always to be switched on; Offline forensics using a backup.db file generated by Line messenger. This research revealed the correlation, interpretation, and analysis of artifacts left in the smartphone. This research also provides guidance for the investigator to solve a real case that using Line messenger as media.

Research about KakaoTalk has developed by J. Choi et al. [19]. Authors perform analysis through a backup database file in KakaoTalk messenger. Methodology used by authors divided into four process as follows: find the backup database file; checked the backup database file was encrypted or not; search for contents loaded from memory in order to determine exact point in time at which to perform decryption on the encrypted backup database files; examining the assembly code AES-128 in CBC mode. This research revealed key generation procedure used by KakaoTalk messenger to encrypt backup file could leak to unauthorized attackers. From this research, we can learn to develop a security mechanism to protect encrypted databases against reverse engineering.

Another research about WeChat messenger by Songyang Wu [20]. In this research, the authors examined data structure and artifact generated by wechat messenger. Authors using Apktool, dex2jar, JD-GUI to perform analysis on wechat messenger. The steps used by the author are grouped and divided into five as follows: installation paths and data acquisition, Decrypting the messages database, Communication records, Moments, and conversion of audio file format. This research revealed that every minor update occurred on WhatsApp may cause changes in the storage structure and the data protection measures. This condition may cause some problems to examiner or investigator to solve the criminal case on wechat messenger. Therefore, the reverse analysis of Android applications, and the data protection mechanism, should be studied continuously to satisfy the new requirements of digital investigation.

2.1. Research Contribution

The significance and novelty of this work, with respect to existing literature, is a procedure for evaluating instant messenger apps in the aspect of nonrepudiation characteristic . This evaluation is essential since it could prevent identity frauds in the development of e-government services through instant messenger in a smart city. This research focuses on the analysis of Line, WhatsApp, and Telegram on the Android, iOS, and Windows Phone operating system to determine the best application in terms of revealing the origin of message data. The procedures in this paper could become guidance for evaluating future release/updates in the instant messenger applications.

3. METHODOLOGY

The outline of research methodology described in Figure 1.

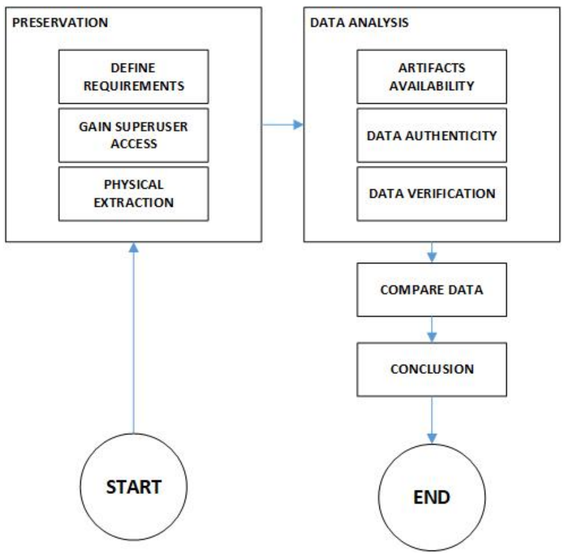


Figure 1 Research methodology

The methodology used for this research is referred to NIST's guideline on mobile device forensics with slight modification to fit the scenario [21].

The first steps to do is Preservation. Which consist of searching for research needs, finding information about gain superuser access for every device used for this research and performing a physical acquisition to retrieve all necessary information and data from the device.

The second step continued with the analysis phase. In this section, the authors conduct in-depth examination obtained data. The examination performed by authors consist of availability checks, data authenticity, and data verification. We check the authenticity and perform data verification by comparing message text data on messenger database with dialogue scenario, checking user's phone number found on contacts database and their userID respectively. After we gain all information about the identity and the message data sent, now we compare obtained data between all messenger on every OS platform, which ones performs the best for identity fraud to help smart city development.

In the last section after all the research objectives have met, we conclude from the research that has been carried out. This conclusion will refer to the most suitable messenger used as a public service media on smart city

4. EXPERIMENT

Before performing the analysis, the authors need to prepare some needs first, based on the requirements in Table 2. We divide the needs of this research into four as follows:

4.1. Grant Superuser Access

For every device, there's a different method to grant superuser access. The difficulty level of granting superuser access determined by OS platform and its device. Every platform we examined needs a different software to gain full physical data of its internal storage.

4.1.1. Android devices (root)

Many people called root to gain access for android superuser folder. In this study authors using twrp/cwm backup to gain root access from xda-developers forum [22]. However, root method used by authors will not work for a different device. There are many tutorials about gain root access we can find on google. Figure 2 is an example of Android superuser folder opened with root explorer.

Table 2: Requirement needs

	Type of Hardware	Device's type	OS version
Hardware	Android Device	Huawei y5ii	Android 5.1
	iOS Device	Iphone 5	iOS 10.3.3
	Windows Phone	Nokia Lumia 730	Windows Phone 8.1
	Android Device	Xiaomi redmi 2	Android 5.1
	Personal Computer	ASUS X455LD	Windows 10
	Type of Software	**Software Brand**	**Application version**
Software	Messenger Application	Line Messenger	8.5
		Telegram Messenger	4.8
		Whatsapp Messenger	2.18
	Analysis Forensics Software	Belkasoft Evidence Center	
		DB SQLite Browser	3.10
		FTK Imager	3.4.3.3
	Type of needs	**Name of Data**	
Additional needs	Email	ridwankresna@gmail.com	
		ridwankresna@yahoo.com	
		ridwan.kresna14@mhs.is.its.ac.id	
	Phone Number	+6282234899191	
		+6289643139991	
		+6289605338880	
	Audio File	*generated on messenger*	
	Image File	FOTO_REKAYASA.jpg	
	Video File	VIDEO_REKAYASA.mkv	
	Scenarios		
Scenarios	Usual application usage without any modification based on dialogue		
	Delete Chat History		

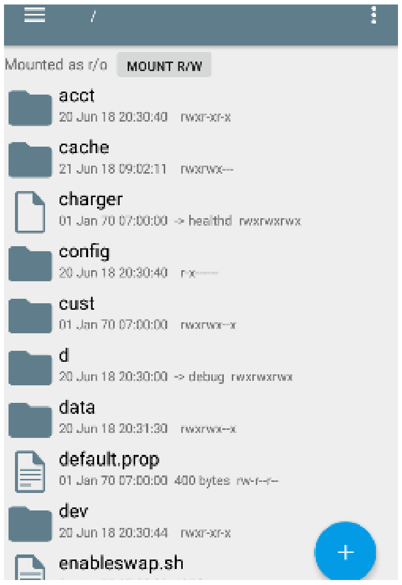

/
Mounted as r/o MOUNT R/W
acct 20 Jun 18 20:30:40 rwxr-xr-x
cache 21 Jun 18 09:02:11 rwxrwx---
charger 01 Jan 70 07:00:00 -> healthd rwxrwxrwx
config 20 Jun 18 20:30:40 r-x------
cust 01 Jan 70 07:00:00 rwxrwx--x
d 20 Jun 18 20:30:00 -> debug rwxrwxrwx
data 20 Jun 18 20:31:30 rwxrwx--x
default.prop 01 Jan 70 07:00:00 400 bytes rw-r--r--
dev 20 Jun 18 20:30:44 rwxr-xr-x
enableswap.sh

Figure 2 Example of rooted Android SU folder

4.1.2. iOS devices (jailbreak)

Authors using iPhone 5 with IOS version 10.3.3. We found that one of the ways to jailbreak this version can be performed by h3lix application that can be found on appvalley website [23]. By installing h3lix, we can gain access to the iOS root folder and get Cydia to install additional application needs to perform next analysis. Figure 3 is an example of superuser folder of iOS opened by fileza app.

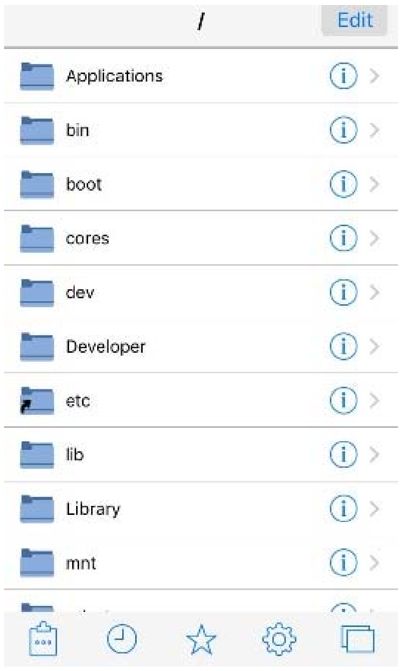

/ Edit
Applications
bin
boot
cores
dev
Developer
etc
lib
Library
mnt

Figure 3 Example of jailbroken iOS SU folder

4.1.3. Windows Phone devices (interop)

On Windows Phone, people called interop to gain SU folder access. There are different methods to perform interop, and not all devices can be interoped. Fortunately, Nokia Lumia 730 is one of the devices that can be interoped although it is so hard to be interoped. Authors using .xap files mods to change default program 'mixradio' on windows phone into an interop application. The difference between interoped and not interoped device is the root access given by device even on PC just like on figure 4.

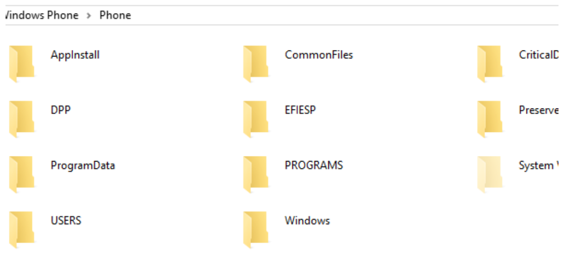

/indows Phone › Phone
AppInstall CommonFiles CriticalD
DPP EFIESP Preserve
ProgramData PROGRAMS System
USERS Windows

Figure 4 Example of Interoped Windows Phone SU folder

4.2. Data Acquisition

There are some ways to get data from a smartphone device, but to get full backup data, need to perform a physical extraction. Physical extraction means copying byte per byte data from smartphone device regardless of system restriction, but it takes so much time to be done. Physical acquisition method used on every platform is different, because not every forensics software supports physical extraction from every platform. Below are physical extraction methods performed by authors for every platform used:

a. Android
For Android device, authors using Android Device Bridge (ADB), busybox (we can find it on playstore) and netcat [24]. The advantage of using busybox and netcat is we do not need an SDCard to copy internal memory physical data from Android smartphone. We can directly transfer physical data from Android smartphones to PC.

b. iOS
For iOS devices or iPhone, authors using Elcomsoft iOS toolkit (I do not have the license) to perform a full physical extraction of iOS. Extraction data stored on C://windows/sysWoW64,

c. Windows Phone
Some software supports windows phone forensics. For this research, we try mobiledit application to perform a physical extraction on Windows Phone.

4.3. Data Analysis Experiment 1

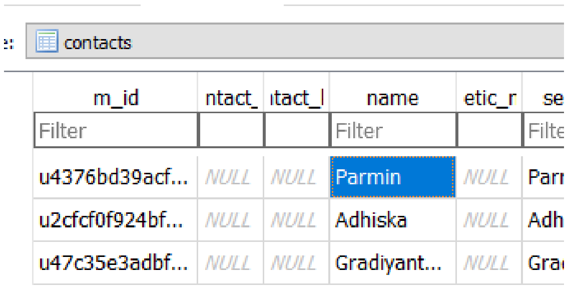
After all data acquired, the next step is data analysis. In this part authors using ftk imager to open and read all of the physical data gained by physical extraction before. Each messenger on each OS platform has different data management, and not every database can be read easily just like SQLite table. Below are explanation and documentation for each messenger on every operating system

4.3.1. Android

The android application generally stores their data on directory \phone\data\. Every application located in the same directory in a different application system folder, so the examiner can easily find the location of the application on Android operating system.

a. Line Messenger on Android Device

Line Messenger located in the directory \phone\data\jp.naver.line. It is using SQLite database format named 'naver_line.' In this database authors using two tables, chat_history and contacts to examine the origin of a message as proof on a real case. From the contacts table, we can get information such as 'name' and 'mid' as a Line user identifier which distinguishes it from the others. Information about contacts table described in Figure 5.

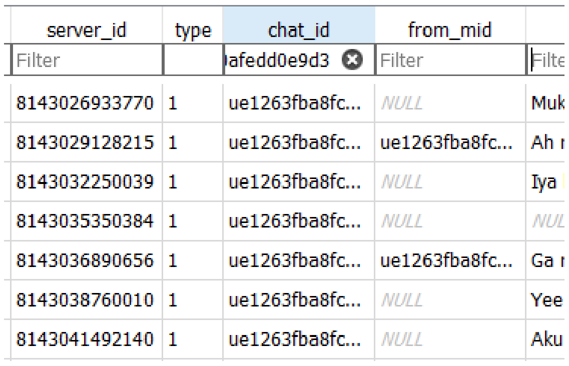

contacts

m_id	ntact_	ıtact_l	name	etic_r	se
Filter			Filter		Filte
u4376bd39acf...	NULL	NULL	Parmin	NULL	Parı
u2cfcf0f924bf...	NULL	NULL	Adhiska	NULL	Adh
u47c35e3adbf...	NULL	NULL	Gradiyant...	NULL	Gra

Figure 5 contacts table from naver_line db Line Android

Meanwhile, on the chat_history table, we can find the complete conversation history of Line user. m_id and server_id column can be used to prove the authenticity of the digital evidence because they are unique and used as an identifier on Line Messenger.

Information about chat_history table described in Figure 6.

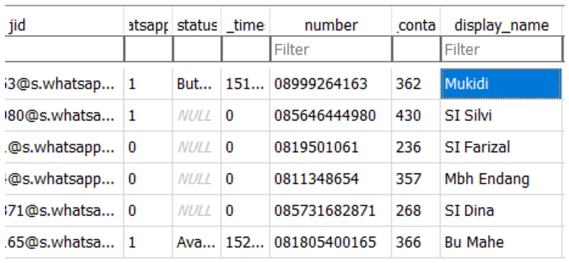


server_id	type	chat_id	from_mid	
Filter		ıafedd0e9d3	Filter	Filte
8143026933770	1	ue1263fba8fc...	NULL	Muk
8143029128215	1	ue1263fba8fc...	ue1263fba8fc...	Ah ı
8143032250039	1	ue1263fba8fc...	NULL	Iya
8143035350384	1	ue1263fba8fc...	NULL	NUL
8143036890656	1	ue1263fba8fc...	ue1263fba8fc...	Ga ı
8143038760010	1	ue1263fba8fc...	NULL	Yee
8143041492140	1	ue1263fba8fc...	NULL	Aku

Figure 6 chat_history table from naver_line db Line Android

b. WhatsApp on Android Device

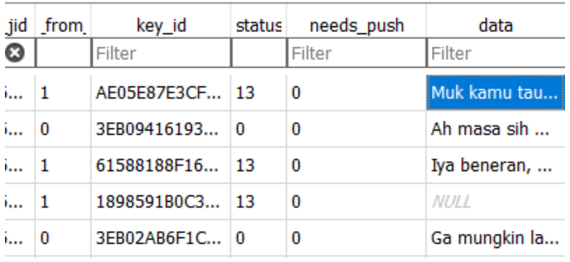
WhatsApp application system located in the directory \phone\data\com.whatsapp. WhatsApp on Android platform using SQLite database format for its data management. Database file used to prove authenticity and the origin of data are wa.db and mgstore.db. From the wa.db file, there is wa_contacts table. We can gain some information from this table such as jid (Whatsapp ID) as an identifier, user's name, and phone number. Information about wa_contacts table described in Figure 7.



jid	ıtsapp	status	_time	number	conta	display_name
				Filter		Filter
ı3@s.whatsap...	1	But...	151...	08999264163	362	Mukidi
ı80@s.whatsa...	1	NULL	0	085646444980	430	SI Silvi
.@s.whatsapp...	0	NULL	0	0819501061	236	SI Farizal
ı@s.whatsapp...	0	NULL	0	0811348654	357	Mbh Endang
ı71@s.whatsa...	0	NULL	0	085731682871	268	SI Dina
.65@s.whatsa...	1	Ava...	152...	081805400165	366	Bu Mahe

Figure 7 wa_contacts table from wa.db WhatsApp Android

Meanwhile on mgstore.db, authors using messages table to find the conversation that has been done based on the dialogue. In mgstore.db we can use a filter of suspect's id on key_remote_id column to make it easy find the conversation data Information about WhatsApp database (mgstore.db) described in Figure 8.

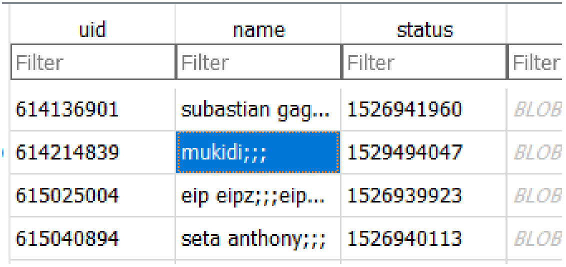


jid	from_	key_id	status	needs_push	data
		Filter		Filter	Filter
i...	1	AE05E87E3CF...	13	0	Muk kamu tau...
i...	0	3EB09416193...	0	0	Ah masa sih ...
i...	1	61588188F16...	13	0	Iya beneran, ...
i...	1	1898591B0C3...	13	0	NULL
i...	0	3EB02AB6F1C...	0	0	Ga mungkin la...

Figure 8 messages table from messages.db WhatsApp Android

c. Telegram on Android Device

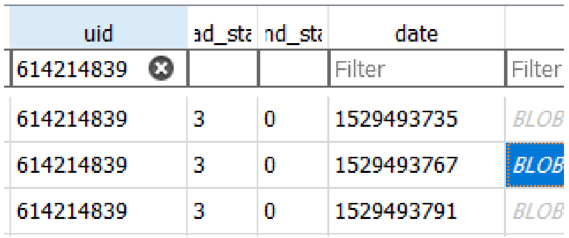
Telegram Android application data can be found in the directory \phone\data\org.telegram.messenger. Telegram Android database named cache4.db. From this database file, authors use users and messages table to examine the origin of a message as proof on a real case. From the contacts database, we can find uid and name of telegram user and also their phone number (encrypted in hexadecimal). Information about Users table described in Figure 9.



uid	name	status	
Filter	Filter	Filter	Filter
614136901	subastian gag...	1526941960	BLOB
614214839	mukidi;;;	1529494047	BLOB
615025004	eip eipz;;;eip...	1526939923	BLOB
615040894	seta anthony;;;	1526940113	BLOB

Figure 9 Users table from cache4.db Telegram Android

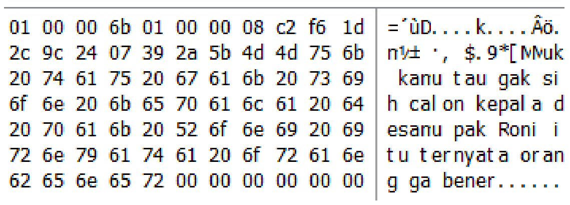
Based on information gained from users table, we can get information about conversation history on messages table. To ease the examination process, we can use the filter on uid to find the dialogue scenario. Information about messages table described in Figure 10.



uid	ad_sta	nd_sta	date	
614214839			Filter	Filter
614214839	3	0	1529493735	BLOB
614214839	3	0	1529493767	BLOB
614214839	3	0	1529493791	BLOB

Figure 10 messages table from cache4.db Telegram Android

Dialogue scenario located on data column which protected by hexadecimal encryption. We can see the detail of the messages sent by using the SQLite browser as in Figure 11.



```
01 00 00 6b 01 00 00 08 c2 f6 1d   =´ùD....k....Âö.
2c 9c 24 07 39 2a 5b 4d 4d 75 6b   n½± ·, $.9*[MMuk
20 74 61 75 20 67 61 6b 20 73 69    kanu tau gak si
6f 6e 20 6b 65 70 61 6c 61 20 64   h cal on kepal a d
20 70 61 6b 20 52 6f 6e 69 20 69   esanu pak Roni  i
72 6e 79 61 74 61 20 6f 72 61 6e   tu ternyat a oran
62 65 6e 65 72 00 00 00 00 00 00   g ga bener......
```

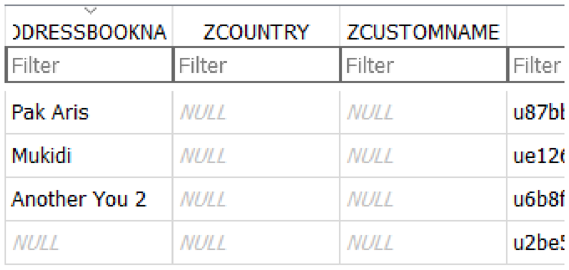
Figure 11 Example of data content on Telegram

4.3.2. iOS

iOS application generally stores their data folder in the directory \private\var\mobile\Containers\Shared\AppGroup\. Every messenger application used in this research located their data in the same directory in a different unique application system folder name. One of the challenge to perform analysis on iOS is we must check every folder on Appgroup folder because none of the apps on iOS provide clues about their folder names like on android - e.g. (com.**whatsapp**).

a. Line Messenger on iOS Device

In my case Line iOS located its data folder in the directory \private\var\mobile\Containers\Shared\AppGroup\E55440B6-C6C2-459C-9C83-3D360A187B32\. It is using the SQLite database format which files named 'line.sqlite.' Tables used in this database to prove identity and origin of messages are Z_MESSAGES and Z_USER. From the Z_User table has the same function as contacts table on Line Android to gain information about user ID and name contacts. Information about Z_User table described in Figure 12.

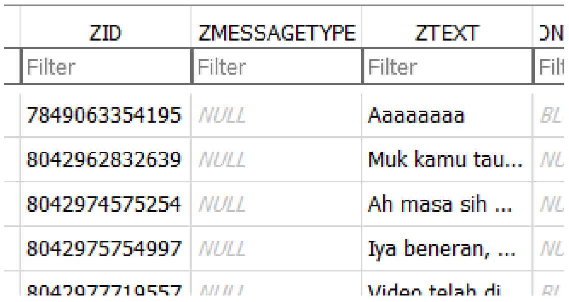


DDRESSBOOKNA	ZCOUNTRY	ZCUSTOMNAME	
Filter	Filter	Filter	Filter
Pak Aris	NULL	NULL	u87bl
Mukidi	NULL	NULL	ue12
Another You 2	NULL	NULL	u6b8f
NULL	NULL	NULL	u2be

Figure 12 Z_USER table from Line.SQLite Line iOS

Meanwhile, Z_MESSAGES table has the same function as chat_history table on Android to gain information about conversation history of Line user.

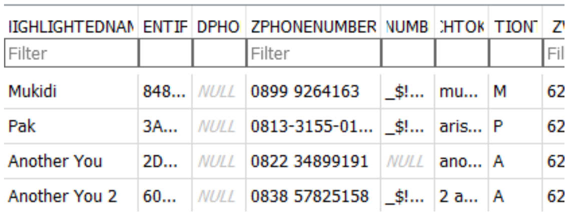
The origin of the message can be examined by checking its ZID / server_id and its timestamp. Information about Z_MESSAGES table described in Figure 13.



ZID	ZMESSAGETYPE	ZTEXT	ZN
Filter	Filter	Filter	Filt
7849063354195	NULL	Aaaaaaaa	BL
8042962832639	NULL	Muk kamu tau...	NU
8042974575254	NULL	Ah masa sih ...	NU
8042975754997	NULL	Iya beneran, ...	NU
8042977719557	NULL	Video telah di	BL

Figure 13 Z_MESSAGE table from Line.SQLite Line iOS

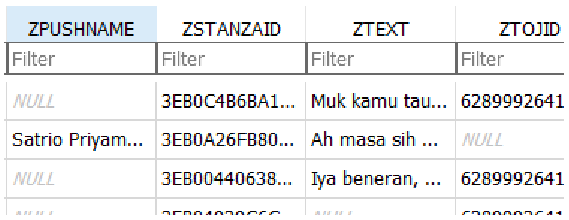
b. WhatsApp on iOS Device
WhatsApp iOS located its data folder in the directory \private\var\mobile\Containers\Shared\AppGroup\55EEB826-A4C9-4D2C-A953 97B9F1CDD08C\. WhatsApp iOS using SQLite database format to store their data. Database files used to prove identity and origin of data are ‘Chatstorage.sqlite’ and ‘ContactsV2.sqlite’. From ContactsV2.sqlite we can gain information about the user’s real name and WhatsApp id on ZWAADRESSBOOKCONTACT table. Information about ZWAADRESSBOOKCONTACT described in Figure 14.



IIGHLIGHTEDNAI	ENTIF	DPHO	ZPHONENUMBER	NUMB	CHTOK	TIONT	Z
Filter			Filter				Fil
Mukidi	848...	NULL	0899 9264163	_$!...	mu...	M	62
Pak	3A...	NULL	0813-3155-01...	_$!...	aris...	P	62
Another You	2D...	NULL	0822 34899191	NULL	ano...	A	62
Another You 2	60...	NULL	0838 57825158	_$!...	2 a...	A	62

Figure 14 ZWAADRESSBOOKCONTACT table from Contacts_v2.sqlite WhatsApp iOS

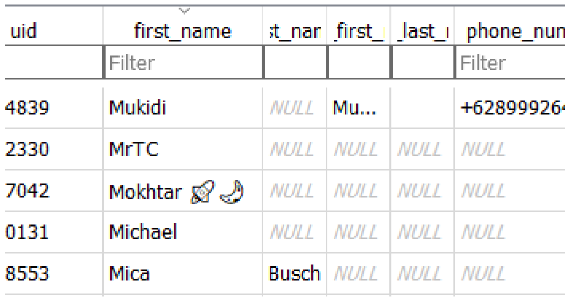
Meanwhile for user’s conversation history located on ZWAMESSAGE table contains information about chat history, the timestamp message sent and ID as identifier chats (ZTANZAID) that can be used as a key to prove the authenticity of conversation. Information about ZWAMESSAGE table described in Figure 15.



ZPUSHNAME	ZSTANZAID	ZTEXT	ZTOJID
Filter	Filter	Filter	Filter
NULL	3EB0C4B6BA1...	Muk kamu tau...	6289992641
Satrio Priyam...	3EB0A26FB80...	Ah masa sih ...	NULL
NULL	3EB00440638...	Iya beneran, ...	6289992641
NULL	[illegible]	NULL	[illegible]

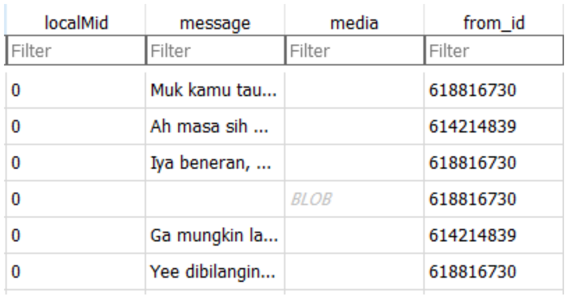
Figure 15 ZWAMESSAGE table from Chatstorage.SQLite WhatsApp iOS

c. Telegram on iOS Device
Telegram iOS application data can be found in the directory \private\var\mobile\Containers\Shared\AppGroup\8668F87B-9DCC-49FD-A1b1-E80209080862\. Database file used to check the origin messages and user’s identity is tgdata.db which located on documents folder. Moreover, the table used in this research are users_v29 and messages_v29. From users_v29 table, we can gain information about the user’s name, their phone number, and uid as an identifier for each user. Meanwhile to check the history of messages and the origin of it we can use messages_v29 table. Authors give a filter on cid column using uid from contacts_v29 table to only gain chat history based on scenario dialogue. Information about the contacts_v29 table and messages_v29 described in Figure 16 and Figure 17 respectively.



uid	first_name	st_nar	first_	last_	phone_nun
	Filter				Filter
4839	Mukidi	NULL	Mu...		+62899926
2330	MrTC	NULL	NULL	NULL	NULL
7042	Mokhtar 🚀 🎺	NULL	NULL	NULL	NULL
0131	Michael	NULL	NULL	NULL	NULL
8553	Mica	Busch	NULL	NULL	NULL

Figure 16 contacts_v29 table from tgdata.db telegram iOS



localMid	message	media	from_id
Filter	Filter	Filter	Filter
0	Muk kamu tau...		618816730
0	Ah masa sih ...		614214839
0	Iya beneran, ...		618816730
0		BLOB	618816730
0	Ga mungkin la...		614214839
0	Yee dibilangin...		618816730

Figure 17 messages_v29 table from tgdata.db telegram iOS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

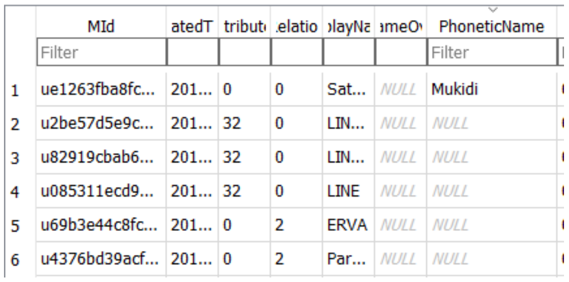
4.3.3. Windows Phone

WhatsApp and Telegram WindowsPhone store their data in the directory \Data\Users\DefApps\APPDATA\Local\Packages\ with different application system folder, meanwhile in this case, Line Windows Phone store its data in the directory \Data\USERS\DefApps\APPDATA\{A18DAAA9-9A1C-4064-91DD-794644CD88E7}.

a. Line Messenger on Windows Phone Device

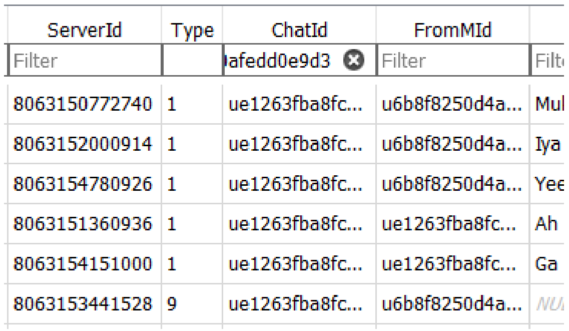
Inside folder {A18DAAA9-9A1C-4064-91DD-794644CD88E7} there is only one SQLite database named Line.db. This database contains every information needed to perform forensic analysis on Line Windows Phones such as user information and history chats. Tables used for this study on Line.db are Contact5 and ChatHistory3. From Contact5 table, we can gain information about Line users id (Mid) to proof the real user identity and their phonetic name on the phonebook.

Meanwhile, from ChatHistory3 table, we can gain information about complete data of user's conversation. m_id and server_id column can be used to prove the authenticity of the digital evidence. Information about Contact5 and ChatHistory3 table described in Figure 18 and Figure 19 respectively.



	MId	atedT	tribut	elatio	olayNa	ameO	PhoneticName
	Filter						Filter
1	ue1263fba8fc...	201...	0	0	Sat...	NULL	Mukidi
2	u2be57d5e9c...	201...	32	0	LIN...	NULL	NULL
3	u82919cbab6...	201...	32	0	LIN...	NULL	NULL
4	u085311ecd9...	201...	32	0	LINE	NULL	NULL
5	u69b3e44c8fc...	201...	0	2	ERVA	NULL	NULL
6	u4376bd39acf...	201...	0	2	Par...	NULL	NULL

Figure 18 Contact5 table from Line.db Line Windows Phone

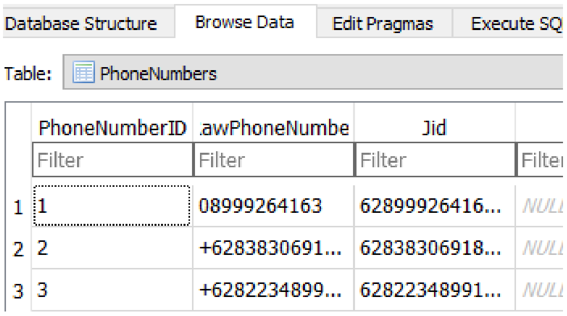


ServerId	Type	ChatId	FromMId	
Filter		afedd0e9d3	Filter	Filt
8063150772740	1	ue1263fba8fc...	u6b8f8250d4a...	Mul
8063152000914	1	ue1263fba8fc...	u6b8f8250d4a...	Iya
8063154780926	1	ue1263fba8fc...	u6b8f8250d4a...	Yee
8063151360936	1	ue1263fba8fc...	ue1263fba8fc...	Ah
8063154151000	1	ue1263fba8fc...	ue1263fba8fc...	Ga
8063153441528	9	ue1263fba8fc...	u6b8f8250d4a...	NU

Figure 19 ChatHistory3 table from Line.db Line Windows Phone

b. WhatsApp on Windows Phone Device

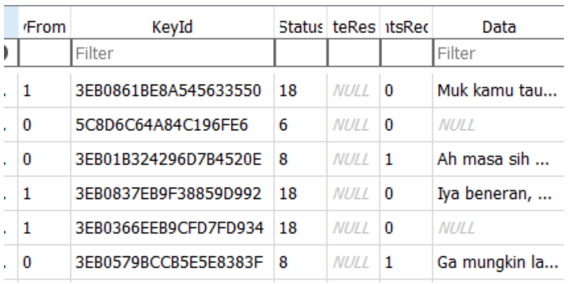
Whatsapp windows phone location full path is \Data\Users\DefApps\APPDATA\Local\Packages\5319725.WhatsApp_cv1g1gvanyjgm\. Databases used to gain information about the origin of data and chat history are contacts.db and messages.db. From contacts.db we can gain information about user's WhatsAppID and their phone number. Information about contacts.db described in Figure 20



	PhoneNumberID	awPhoneNumbe	Jid	
	Filter	Filter	Filter	Filter
1	1	08999264163	62899926416...	NULL
2	2	+6283830691...	62838306918...	NULL
3	3	+6282234899...	62822348991...	NULL

Figure 20 PhoneNumbers table from contacts.db WhatsApp Windows Phone

To get complete information about chat history authors examines Messages table on ChatStorage.sqlite database. Information gathered are unique KeyId as an identifier to prove the authenticity of a message, data content and the timestamp message sent. Information about ChatStoreage.sqlite described in Figure 21.

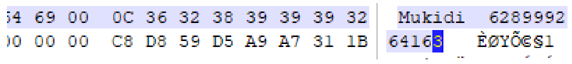


From	KeyId	Status	teRes	itsRec	Data
	Filter				Filter
1	3EB0861BE8A545633550	18	NULL	0	Muk kamu tau...
0	5C8D6C64A84C196FE6	6	NULL	0	NULL
0	3EB01B324296D7B4520E	8	NULL	1	Ah masa sih ...
1	3EB0837EB9F38859D992	18	NULL	0	Iya beneran, ...
1	3EB0366EEB9CFD7FD934	18	NULL	0	NULL
0	3EB0579BCCB5E5E8383F	8	NULL	1	Ga mungkin la...

Figure 21 Messages table from ChatStorage.sqlite WhatsApp Windows Phone

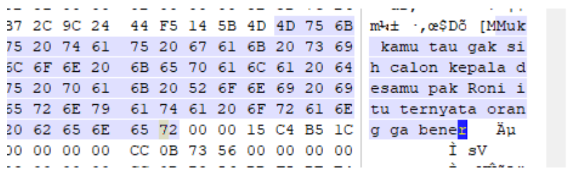
c. Telegram on Windows Phone Device

Telegram Windows phone store its data in the directory \Data\Users\DefApps\APPDATA\Local\Packages\TelegramMessengerLLP.TelegramMessengerBeta_t4vj0pshhgkwm. There is no SQLite database here, because telegram database on windows phone built on .dat files format. Authors using winhex to gain information from this kind of database. Files analyzed from telegram windows phones are dialogs.dat and users.dat. From Users.dat we gained information about the user and their phone number Information about Users.dat and dialogs.dat described in Figure 22 and 23 respectively. From dialogs.dat we gained information about the conversation, but we can only determine the content not the sender and recipient of the content



```
54 69 00  0C 36 32 38 39 39 39 32 | Mukidi  6289992
00 00 00  C8 D8 59 D5 A9 A7 31 1B | 64163   ÈØYÕ€§1
```

Figure 22 Users.dat Telegram Windows Phone



```
37 2C 9C 24  44 F5 14 5B 4D 4D 75 6B | m¼±  ·,œ$Dõ [MMuk
75 20 74 61  75 20 67 61 6B 20 73 69 |  kamu tau gak si
6C 6F 6E 20  6B 65 70 61 6C 61 20 64 | h calon kepala d
75 20 70 61  6B 20 52 6F 6E 69 20 69 | esamu pak Roni i
65 72 6E 79  61 74 61 20 6F 72 61 6E | tu ternyata oran
20 62 65 6E  65 72 00 00 15 C4 B5 1C | g ga bener   Äµ
00 00 00 00  CC 0B 73 56 00 00 00 00 |          Ì sV
```

Figure 23 dialogs.dat Telegram Windows Phone

4.3.4. Undiscoverable Artifacts Media

All media can be found and proofed its authenticity from the database except for Videos on Line Android and Line Windows Phone. There is no video media found from scenario dialogue except its thumbnail. After doing some trial error examination, we found that videos on these two platforms only can be accessed if device online. When the device is offline / restarted, video files will be deleted. So it is hard to prove any video evidence from Line Android and Line Windows Phone. For discoverable media will be explained in Table 3. Moreover, Table 4,5 and 6 describes the summary of experiments results.

4.4. Data Analysis Experiment 2

Experiment 2 is to destroy the user's chat history. After we perform the second experiment, we experience some data loss in the chat history and several media, but some messenger pretends to keep its media files although there is a notification that the data has deleted when we delete the chat history.

5. CONCLUSION

Physical extraction used by the authors is the best way to gather all information from any devices. This method can cover almost any messenger's artifacts on every OS platform in this study. However, if the application does not keep their data on storage as a cache or offline data, this method will be useless just like happened on video evidence on Line Android and Windows phone. Nevertheless, anything about identity fraud and origin of the data can be proofed by a unique message identifier from each messenger on every platform. Every messenger application has a different column to store the data about message identifier, but they have the same value for each user and unique each other. We can also detect every messenger user by knowing their user id on the contacts table.

From this research WhatsApp messenger on every OS platform has the best performance. WhatsApp not only identifies user's id but also their phone number like Telegram, in addition, WhatsApp has no encryption on their conversation data that make examiner easier to solve a case on WhatsApp than Telegram. WhatsApp also manages their media data well, all media have names registered in the database, have an extension and neatly organized in the media folder. Afterward, this research still has a weakness. The weakness of this research is the result may change with significant updates of the messenger application. Messenger applications are often updated for some reason. This update could change the path folder, database format or anything that will make a different way to obtain artifacts. This is a challenge for the examiner to perform more analysis forensics on the latest version of application messenger to make it observable and could be examined.

ACKNOWLEDGMENT

This research was supported/funded by "Penelitian Terapan Unggulan Perguruan Tinggi 2018" research grant, under contract number: 998/PKS/ITS/2018.

REFERENCES:

[1] E. Siregar, "Aplikasi Berbalas Pesan Terpopuler di Indonesia," Liputan 6, 08 November 2017. [Online]. Available: http://news.liputan6.com/read/3154722/aplikasi-berbalas-pesan-terpopuler-di-indonesia. [Accessed 5 June 2018].

[2] K. A. Rizqo, "Namanya Dicatut di Kasus Penipuan Via WhatsApp, Imam Lapor Polisi," detiknews, 14 September 2017. [Online]. Available: https://news.detik.com/berita/d-3642076/namanya-dicatut-di-kasus-penipuan-via-whatsapp-imam-lapor-polisi. [Accessed 5 June 2018].

[3] A. A. Pangera and D. Ariyus, Sistem Operasi, Surabaya: CV ANDI OFFSET, 2005.

[4] statcounter, "Operating system market share," January 2018. [Online]. Available: http://gs.statcounter.com/os-market-share. [Accessed 5 June 2018].

[5] J. Sylve, A. Case, L. Marziale and G. G. Richard, "Acquisition and analysis of volatile memory from android devices," Digital Investigation, vol. 8, no. 3-4, pp. 175-184, 2012.

[6] V. Thing, K.-Y. Ng and E.-C. Chang, "Live memory forensics of mobile phones," Digital Investigation, no. 7, pp. S74-S82, 2010.

[7] E. S. Canlar, M. Conti, B. Crispo and R. D. Pietro, "Windows Mobile LiveSD Forensics," Journal of Network and Computer Applications, no. 36, pp. 677-684, 2013.

[8] J. Gregorio, A. Gardel and B. Alarcos, "Forensic analysis of Telegram Messenger for Windows Phone," Digital Investigation, no. 22, pp. 88-106, 2017.

[9] N. A. Aziz, F. Mokti and M. N. M. Nozri, "Mobile Device Forensics: Extracting and Analysing Data from an Android-based Smartphone," Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic, no. 32, pp. 123-128, 2015.

[10] C. Anglano, M. Canonico and M. Guazzone, "Forensic analysis of Telegram Messenger on Android smartphones," Digital Investigation, vol. 23, pp. 31-49, 2017.

[11] M. Kaart and S. Laraghy, "Android forensics: Interpretation of timestamps," Digital Investigation, vol. 11, pp. 234-248, 2014.

[12] A. Hoog, Android Forensics: Investigation, Analysis, and Mobile Security for Google Android, Waltham: Syngress, 2011.

[13] M. I. Husain and R. Sridhar, "iForensics: Forensic Analysis of Instant Messaging on Smart Phones," in International Conference on Digital Forensics and Cyber Crime, 2009.

[14] O. S. Adebayo, S. A. Sulaiman, O. Osho, S. M. Abdulhamid and J. K. Alhassan, "FORENSIC ANALYSIS OF KIK MESSENGER ON ANDROID DEVICES," International Engineering Conference, 2017.

[15] K. M. Ovens and G. Morison, "Forensic analysis of Kik messenger on iOS devices," Digital Investigation, vol. 17, pp. 40-52, 2016.

[16] C. Sgaras, T. Kechadi and N.-A. Le-Khac, "Forensics Acquisition and Analysis of Instant Messaging and VoIP Applications," in International Workshop on Computational Forensics, Tsukuba, 2014.

[17] G. B. Satrya, P. T. Daely and Y. S. Shin, "Android Forensics Analysis: Private Chat on Social Messenger," in International Conference on Ubiquitous and Future Networks 2016 (ICUFN 2O16), Vienna, 2016.

[18] G. B. Satrya and M. A. Nugroho, "Digital forensics study of internet messenger: Line artifact analysis in Android OS," in International Conference on Control, Electronics, Renewable Energy, and Communications, 2016.

[19] J. Choi, J. Park and H. Kim, "Forensic analysis of the backup database file in KakaoTalk messenger," BigComp, vol. 17, pp. 156-161, 2017.

[20] S. Wu, X. Xiong, Y. Zhang, X. Wang and L. Du, "Forensic analysis of WeChat on Android smartphones," DFRWS, vol. 21, pp. 3-10, 2016.

[21] R. Ayes, S. Brothers and W. Jansen, "http://www.mobileforensicscentral.com," May 2014. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-101r1.pdf. [Accessed 9 June 2018].

[22] "[ROOT] [TWRP] [CWM] Huawei Y5 II CUN-L21 / CUN-L03 / ?," [Online]. Available: https://forum.xda-developers.com/huawei-y5/how-to/root-huawei-y5-ii-cun-l21-l03-t3536329. [Accessed 7 June 2018].

[23] "appvalley," [Online]. Available: https://appvalley.vip/app/. [Accessed 07 June 2018].

[24] J. I. James, "Imaging Android with ADB, Root, Netcat and DD," [Online]. Available: https://dfir.science/2017/04/Imaging-Android-with-root-netcat-and-dd.html. [Accessed 8 June 2018].

[25] "root," [Online]. Available: https://www.xda-developers.com/root/.

[26] "Hướng dẫn interop winphone 8.1 vĩnh viễn," winphoneviet, [Online]. Available: http://www.winphoneviet.com/forum/threads/huong-dan-interop-winphone-8-1-vinh-vien.184391/. [Accessed 9 june].

Table 3 Database artifact for each Messenger and Operating System

Operating System	Messenger	Database
Android	Line	\data\jp.naver\line\databases\naver_line
	WhatsApp	\data\com.whatsapp\databases\mgstore.db
		\data\com.whatsapp\databases\wa.db
	Telegram	\data\org.telegram.messenger\cache4.db
iOS	Line	\private\var\mobile\Containers\Shared\AppGroup\E55440B6-C6C2-459C-9C83-3D360A187B32\Library\Application Support\Private Store\p_u96f6579185e5abe1a1aaa5203a3da41b\Library\Messages\Line.sqlite
	WhatsApp	\private\var\mobile\Containers\Shared\AppGroup\55EEB826-A4C9-4D2C-A953 97B9F1CDD08C\ChatStorage.sqlite
		\private\var\mobile\Containers\Shared\AppGroup\55EEB826-A4C9-4D2C-A953 97B9F1CDD08C\ContactsV2.sqlite
	Telegram	\private\var\mobile\Containers\Shared\AppGroup\8668F87B-9DCC-49FD-A1b1-E80209080862\Documents\tgdata.db
Windows Phone	Line	\Data\USERS\DefApps\APPDATA\{A18DAAA9-9A1C-4064-91DD-794644CD88E7}\ Local\Line.db
	WhatsApp	\Data\Users\DefApps\APPDATA\Local\Packages\5319725.WhatsApp_cv1g1gvanyjgm\Local State\messages.db
		\Data\Users\DefApps\APPDATA\Local\Packages\5319725.WhatsApp_cv1g1gvanyjgm\Local State\contacts.db
	Telegram	\Data\Users\DefApps\APPDATA\Local\Packages\TelegramMessengerLLP.TelegramMessengerBeta_t4vj0pshhgkwm\LocalState\dialogs.dat
		\Data\Users\DefApps\APPDATA\Local\Packages\TelegramMessengerLLP.TelegramMessengerBeta_t4vj0pshhgkwm\LocalState\users.dat

Table 4 Media Artifacts for each Messenger and Operating Systems

OS	Messenger	Media Path	
Android	Line	Video	Not found
		Voice Notes	\SDCard\Android\data\jp.naver.line.android\storage\mo
		Images	
	WhatsApp	Video	\SDCard\whatsapp\media\WhatsApp Video\
		Voice Notes	\SDCard\whatsapp\media\WhatsApp Voice Notes\
		Images	\SDCard\whatsapp\media\WhatsApp Images\
	Telegram	Video	\SD Card\Telegram\Telegram Video
		Voice Notes	\SD Card\Telegram\Telegram Audio
		Images	\SD Card\Telegram\Telegram Images
iOS	Line	Video	\private\var\mobile\Containers\Data\Application\ BCA0611F-A443-4529-A63B-77F6E0734251\Library\Chaces\ChaceableVideo\
		Voice Notes	\private\var\mobile\Containers\Data\Application\BCA0611F-A443-4529-A63B-77F6E0734251\Library\Application Suuport\PrivateStore\P_u96f6579185e5abe1a1aaa5203a3da41b\Message Attachments.
		Images	
	WhatsApp	Video	\private\var\mobile\Containers\Shared\AppGroup\55EEB826-A4C9-4D2C-A953-97B9F1CDD08C\Message\Media.
		Voice Notes	
		Images	
	Telegram	Video	\private\var\mobile\Containers\Shared\AppGroup\8668F87B-9DCC-49FD-A1b1-E80209080862\Documents\files
		Voice Notes	
		Images	
Windows Phone	Line	Video	Not Found
		Voice Notes	\Data\USERS\DefApps\APPDATA\{A18DAAA9-9A1C-4064-91DD-794644CD88E7}\Local\Cache
		Images	
	WhatsApp	Video	\Data\USERS\Public\Pictures\WhatsApp
		Images	
		Voice Notes	\Data\USERS\Public\Pictures\WhatsApp\PTT
	Telegram	Video	\data\users\DefApps\AppData\Local\Packages\TelegramMessengerLLP.TelegramMessengerBeta_t4vj0pshhgkwm
		Voice Notes	
		Images	

Table 5 comparison of all messenger application data

	Android			iOS			Windows Phone		
	Line	WhatsApp	Telegram	Line	WhatsApp	Telegram	Line	WhatsApp	Telegram
Conversation Database	naver_line	mgstore.db	cache4.db	Line.sqlite	ChatStorage.sqlite	tgdata.db	line.db	messages.db	dialogs.dat
Contacts Database		wa.db			ContactsV2.sqlite			contacts.db	users.dat
Contacts Table	Contacts	wa_contacts	users	ZUSER	ZWAADRESSBOOKCONTACT	users_v29	Contact5	PhoneNumbers	Inside users.dat file
Unique Conversation Identifier	server_id	key_id	-	ZID	ZTANZAID	-	ServerId	KeyId	Unidentified
User ID	mid	jid	uid	ZMID	ZWHATSAPPID	uid	MId	Jid	Unidentified
Timestamp type	Unix epoch time	Unix epoch time	Unix epoch time	Unix epoch time	Cocoa core time	Unix epoch time	Human Date	Unix epoch time	Unidentified
conversation table	chat_history	messages	messages	ZMESSAGE	ZWAMESSAGE	messages_v29	ChatHistory3	Messages	Inside dialogs.dat
Text Encryption	No	No	Yes	No	No	No	No	No	No
Media Folder	separated in different location	separated in different location	separated in different location	separated in different location	Separated in the same application data folder	Separated in the same application data folder	Separated in the same application data folder	separated in different location	In the same location with database
Media File Extension									
Video	Media not found	.mp4	.mp4	.mp4	.mp4	.mov	Media not found	.mp4	.mp4
Image	raw file (no extension)	.jpg	.jpg	.jpg	.jpg	.jpg	raw file (no extension)	.jpg	.jpg
Voice Note	.aac	.opus	.ogg	.mp4	.opus	raw file (no extension)	raw file (no extension)	.opus.waptt	.wav

Table 6 Experiments 2 Result

Operating System	Messenger	Chat History	Media		
			Video	Images	Voice Notes
Android	Line	Deleted	Not found	Only Thumbnail	Deleted
	WhatsApp	Deleted	Deleted	Deleted	Deleted
	Telegram	Deleted	Available	Available	Available
iOS	Line	Deleted	Available	Deleted	Deleted
	WhatsApp	Deleted	Deleted	Deleted	Deleted
	Telegram	Deleted	Only Thumbnail	Deleted	Deleted
Windows Phone	Line	Deleted	Deleted	Deleted	Deleted
	WhatsApp	Deleted	Available	Available	Deleted
	Telegram	Deleted	Available	Available	Available