The International Journal Of Engineering And Science (IJES)
|| Volume || 6 || Issue || 1 || Pages || PP 82-92|| 2017 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

Proposed Workable Process Flow with Analysis Framework for Android Forensics in Cyber-Crime Investigation

Naing Linn Htun[1], Mie Mie Su Thwin[2]
[1]Cyber Security Lab, University of Computer Studies, Yangon, Myanmar
[2]Associate Professor, University of Computer Studies, Yangon, Myanmar

ABSTRACT

Nowadays, android smartphones are becoming more popular and the greatest platform for mobile devices which has capability to run millions of mobile phones in about more than 200 countries. It may bring not only convenience for people but also crimes or security issues. Some people are committed the crimes by using the technology and mobile devices. So, android forensics is very important and necessary in cyber-crime investigation. With no doubt, this proposed process flow and framework will definitely support for android forensics in developing countries' cyber-crime investigation. Because it provides to solve the crimes with applicable guidelines and includes Open Source Tools, Linux command-line utility, Android Debug Bridge (ADB) commands, Freeware tools and Proposed tools. Although forensics tools are non-commercial in this framework, they can cover and support for android forensics process.

Keywords: *Android Live Forensics, Static Forensics, Evidences Data Collection, Linux Command-Line Utility, Forensics Process Flow.*

Date of Submission: 10 January 2017 Date of Accepted: 26 January 2017

I. INTRODUCTION

Android smartphones are rapidly developed than any other consumer technology in the world. They involve confidential and differences of data; from normal to professional life. Most people use the smartphones for their business or personalities but some people use for their profits by illegally. So, some researchers and forensic organizations have been searching the different ways to solve the crimes and they proposed the guidelines and frameworks for android forensics in cyber-crime investigation.

In 2006, National Institute Standards and Technology (NIST) pointed out that its proposed Forensic guidelines and procedures should be consistent with the organizations policies and all applicable laws. Organizations should include investigators, technical experts and legal advisors in the development of guidelines and procedures as a quality assurance measure [1]. In 2014, NIST provides basic information on mobile forensics tools and five stages for investigation process that are preservation, acquisition, examination, analysis, and reporting [2]. And then most of the researchers proposed only process flow or framework for android forensics.

This proposed process flow with framework based on standard forensics guidelines and procedures. In this process flow, it has seven stages and framework will performing to Examination and Analysis stage. Preparation, Examination and Analysis, Review – These three stages are working in forensics lab. Determine Scope of Crime Scene and Secure Evidence Devices Collection – These two stages are working in crime location. Presentation stage is for court and Documentation and Preservation stage is need from start to end. In this framework, it divided two sections are Live and Static forensics which include bypass lock screen, evidence data collection (volatile and non-volatile) and data acquisition.

This proposed work aimed to develop the android investigation processes for forensic organizations in developing countries. Some of the developing countries cannot afford the expensive tools for android investigation. In this framework, it used Linux command-line utility, ADB commands, Open Source Tools and Proposed tools. So, it is very sensible for forensics organizations in developing countries. And it is very extensible and they can improve it with their new tools or new processes.

II. BACKGROUND

Android is a mobile operating system that is based on a modified version of Linux. The main components of Android are Linux kernel, Android Libraries, Application framework and Applications. Linux kernel contains all the low-level device drivers for the various hardware components of an Android device. Android Libraries contain all the code that provides the main features of an Android OS. Android runtime provides a set of core libraries that it includes Dalvik virtual machine or ART Android Runtime [3].

DOI : 10.9790/1813-0601028292

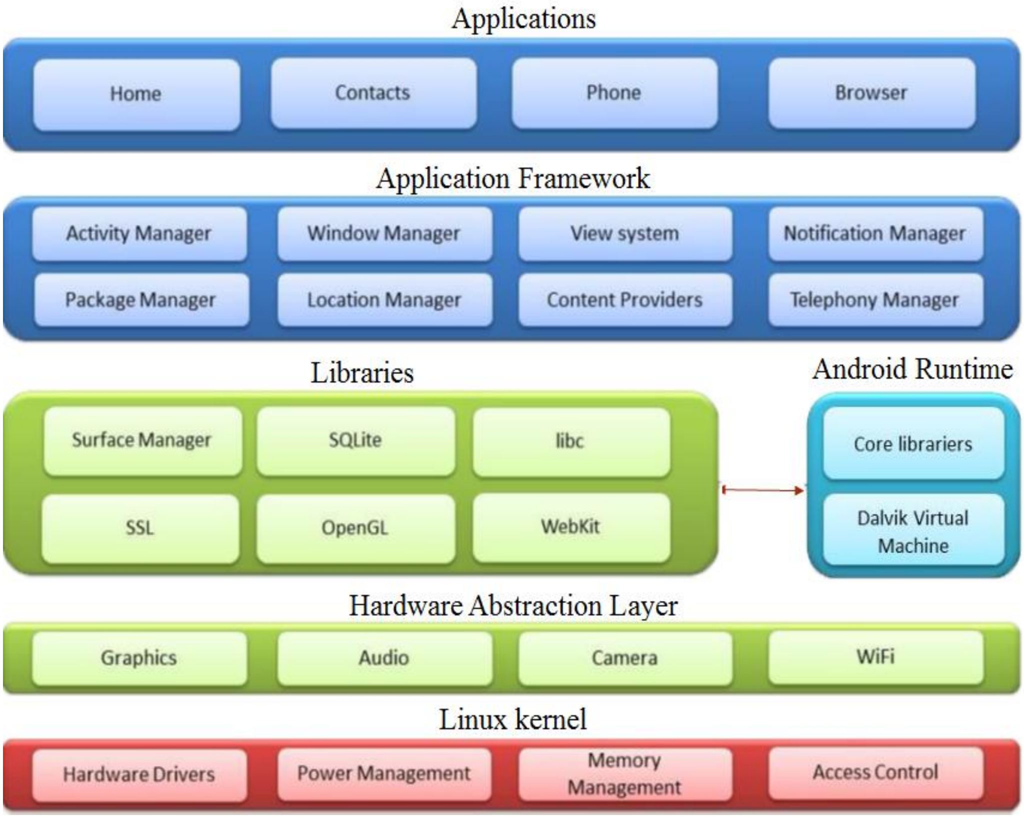


Fig 1 Android Software Stack

Android flash storage is split into several partitions, such as /system for the operating system itself, and /data for user data and application installations [4]. It need to know about following partitions for android investigation.

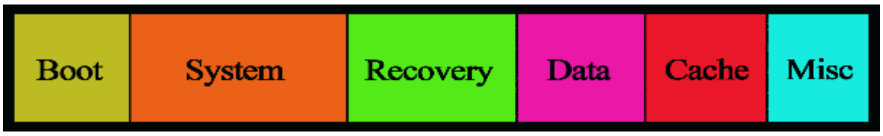


Fig 2 Android partition details

III. RELATED WORK

For our proposed work, Forensics process flow and framework developed in each digital forensics field are learned through the following research papers.

In [6], they proposed smartphone forensics model for window mobile device. It includes twelve stages and compare with some investigation models. But they emphasize on the specific information flow associated with the forensic investigation of Windows mobile devices. In ANDROPHSY forensic framework system [7], the first step is to create a case for the incident. Every user is assigned a role. The user roles defined in the framework are analyst, investigator and administrator. The system considered the forensics investigation process like as NIST. It includes Data Collection, Examination, Analysis and Reporting. It used low level Linux and Android built-in forensic functionalities such as *dd* and *adb* commands. ANDROPHSY compared evaluation result with other tools.

ISAK MRKAIĆ [5] presents open source-free tools and to illustrate how to forensically recover data from Android based devices. In this paper, demonstrate the Andriller tool for unlock pattern, NowSecure Forensics tool for logical acquisition of data and some tools for other processes. In [8], they purposed to demonstrate the process of digitally investigating on android mobile devices and then build up a digital forensic framework for investigating Android smartphones. In this framework, it includes nine steps of evidence extraction process for mobile forensics. And this paper recommends the ADB tool for forensics case and the Lowmanio Foreman tool for other management case. In [9], they showed the rooting process with SRS tool on Samsung S3 phone. And then they created an image of phone partition with *.dd* file. Finally, they extracted the messages for Viber on trial version of UFED Physical analyzer.

In [10], they proposed evidence collection and analysis methodology for android devices. In this paper, the major contribution is an in-depth evidence collection and analysis methodology for forensic practitioners. Proposed methodology based on four phases of Cloud Forensic Framework (Martini & Choo 2012). The first phase is Identification and Preservation. Second phase is setup Bootloader for Live OS and setup Live OS in Memory to collect physical image of device partitions. Third phase is Examination and Analysis, it examines App files in private and external storage, App databases, Account data and analyze App. The last phase is Reporting and Presentation.

Andri P. Heriyanto [11] revealed the proper procedure for acquiring the volatile memory in the Android smartphone. This paper describes briefly workflow of L.Simao et al., 2011 and discusses the use of Linux Memory Extraction (LiME) for dumping the volatile memory. It also discusses the analysis process of the memory image with Volatility 2.3.

IV. PROPOSED WORKABLE PROCESS FLOW FOR ANDROID FORENSICS INVESTIGATION

In cyber-crime investigation, many process flows or models already proposed for digital forensics and analysis. But anyone has not proposed process flow with framework for android forensics investigation yet. This process flow based on standard forensics process models and their main four stages are Collection, Examination, Analysis and Reporting. But this process flow has seven stages and update action and it can trace or loop to completed. Although this process flow emphasizes the android forensics investigation, it may be flexible for other digital forensics investigation.

Preparation: In the first Stage, it need to prepare Techniques and Tools, Knowledge, Plans and etc. before going to crime scenes. Knowledge that is understanding and knowing the nature of crimes and activities. And android mobile technology is rapidly changing and growing every year with various mobile device models, firmware versions, features, accessories and so on. So, forensics team will train to their members based on previous crime reviews, sharing experiences and new technologies with discussion seminars. And then they will be planning to minimize the risk for future investigation. And they also need to prepare tools that not only forensics hardware/software tools but also packing materials for evidence devices.

Determine Scope of Crime Scene: This stage is preventing the unauthorize person to access of the crime scene. If anyone can pass in crime scene, the evidence things can be lost or destroyed. And mobile devices are more easy to take and hide because they are very portable and smaller than other digital devices. Otherwise, anyone can make at least switch off power or factory rest. It may be lost evidences data on mobile devices.

Secure Evidence Devices Collection: This stages is especially to get volatile data when investigator collects the evidence devices. Because volatile data easy to lose and difficult to keep. If mobile devices power is on, (i) Check the battery percentage and if it is very low, recharging the evidence devices. (ii) If mobile device is connected with computer, check the any process is running and kill/close all it (or) wait to finish. And then remove the USB cable. (iii) Criminal can control the evidence devices via Phone Call, Message, Bluetooth, Wireless or Mobile Data Network. So, it need to prevent any external control to the evidence devices (eg. Faraday Bag). Otherwise, Airplane Mode can disconnect the devices from any communications.

Documentation and Preservation: This stage is recording all stages and labelling all evidence devices to transfer to forensics workshop. For recording, all of investigation activities are recorded from crime scene to done. It includes photos, circumstances surrounding the incident, locations and crime scene mapping, existing states of evidence devices, people lists and etc. After collecting the evidence devices, it can be identification and labeling to them. And then need to check again each evidence devices for safety in packing and transportation to forensics department.

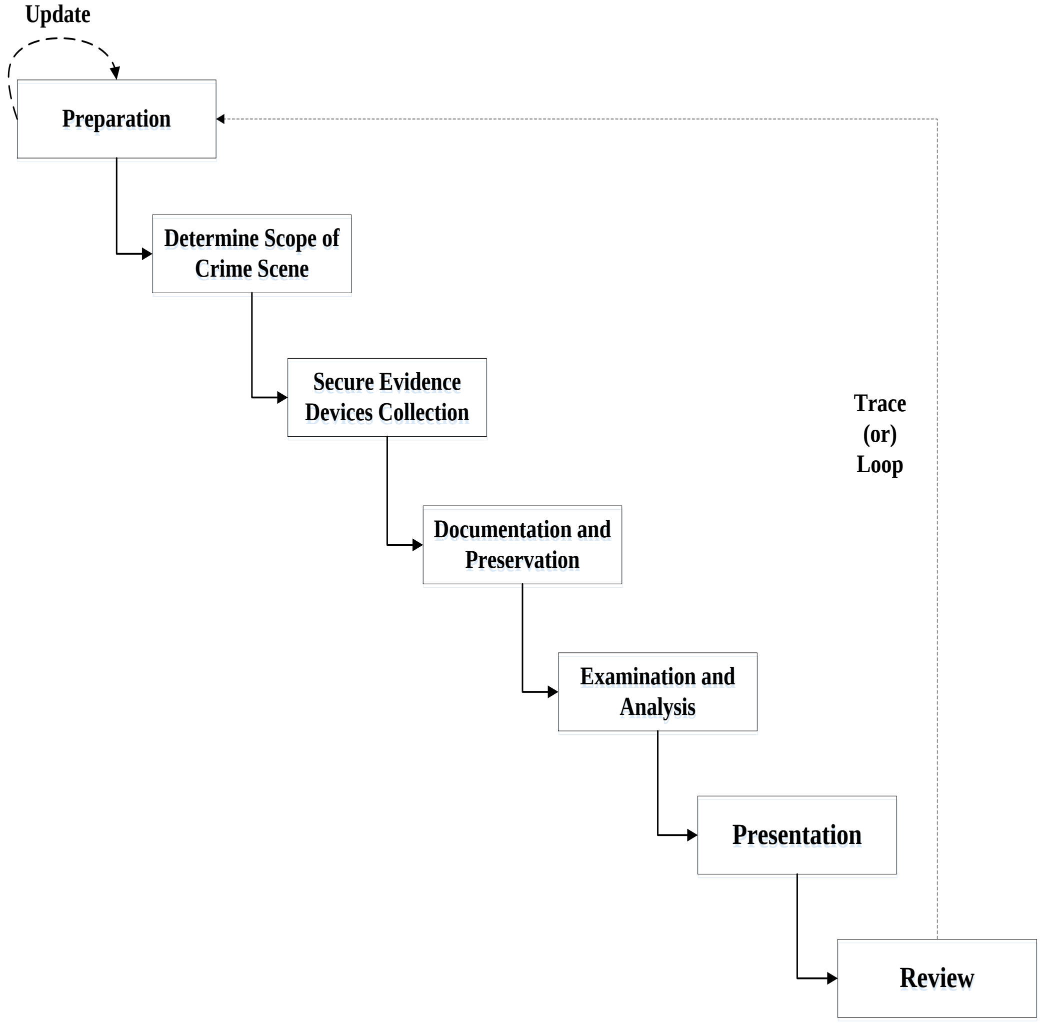

Update
Preparation
Determine Scope of Crime Scene
Secure Evidence Devices Collection
Documentation and Preservation
Examination and Analysis
Presentation
Review
Trace (or) Loop

Fig 3 Proposed workable process flow for android forensics investigation

Examination and Analysis: This stage is extract the data on evidence devices, and then separate data types and levels of data such as volatile data, active data, recovery data, hidden data, data with password, mobile profile data, non-ASCII and etc. And then analyzing the all of data types and identifying relationships between their types.

Presentation: This stage is reporting and presenting to court of law. So, it needs to build a chain of custody documents after the whole analysis. It includes all of evidences data, photos, materials and etc. to present in court.

Review: In the last stage is completely review on entire forensics process such as all of activities, stages and etc. And this review will be useful and support for preparation of future investigations. Forensics Team can be tracing, studying, preparing, training, discussing and etc.

V. PROPOSED ANALYSIS FRAMEWORK FOR ANDROID FORENSICS INVESTIGATION

This framework is performing the Examine and Analysis stage in proposed process flow. And update of Preparation stage is one of the process to support for proposed framework. This framework involves Live forensics and Static forensics as digital forensics nature. But it needs to consider the Lock Screen Mode for android forensics investigation. Criminals may be locked by pattern or pin code or password their android smartphone. So, investigator needs to get fully access to investigate on evidence devices. After examination and analysis, their result data can be supported to build chain of custody and reported to court for prosecute.

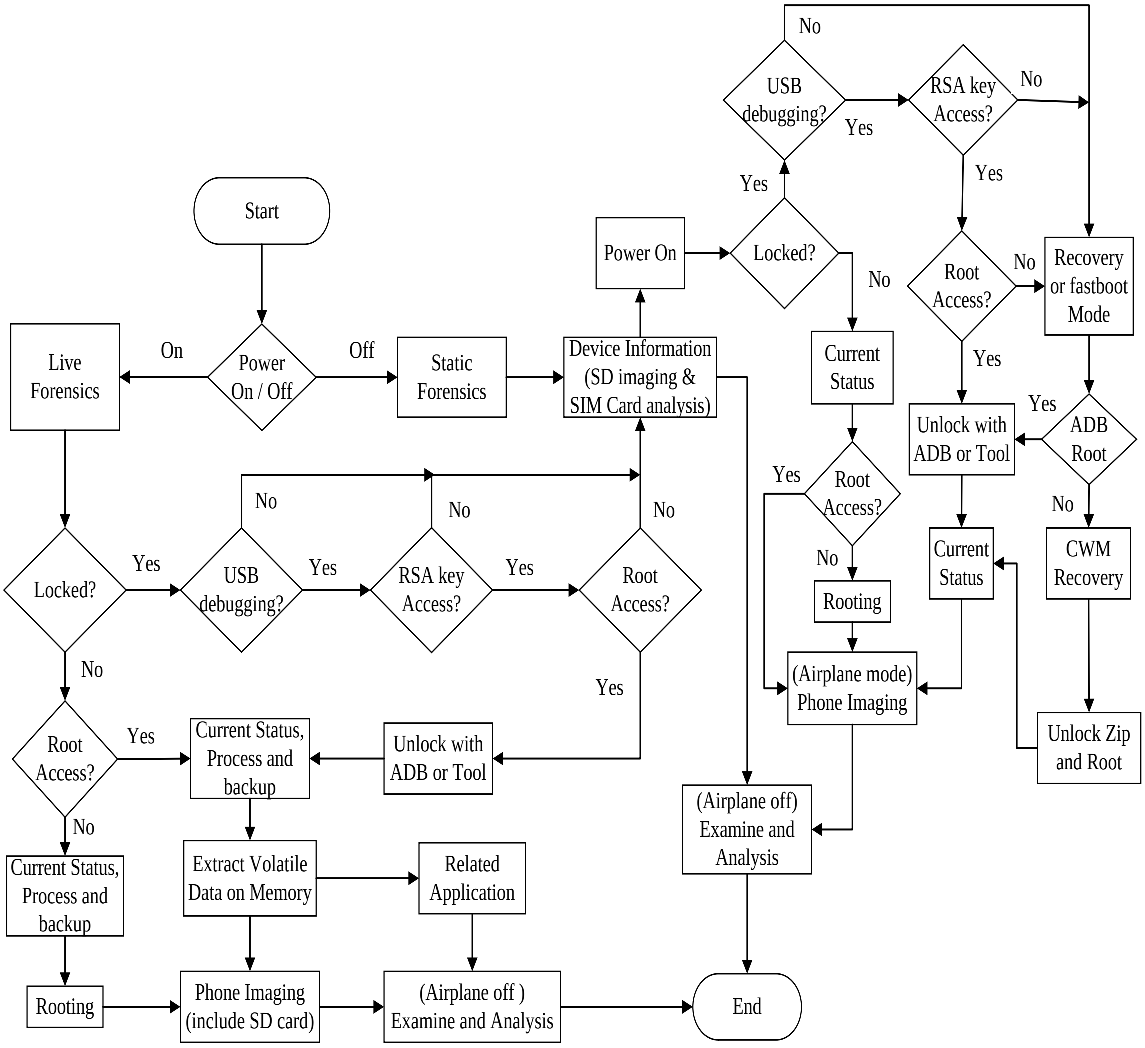

Start
Power On / Off
On
Off
Live Forensics
Static Forensics
Device Information (SD imaging & SIM Card analysis)
Power On
Locked?
USB debugging?
RSA key Access?
Root Access?
Recovery or fastboot Mode
ADB Root
Unlock with ADB or Tool
CWM Recovery
Unlock Zip and Root
Current Status
Rooting
(Airplane mode) Phone Imaging
(Airplane off) Examine and Analysis
Current Status, Process and backup
Extract Volatile Data on Memory
Related Application
Phone Imaging (include SD card)
(Airplane off) Examine and Analysis
End
Yes
No

Fig 4 Proposed analysis framework for android forensics investigation

Lock Screen Mode on Live Forensics: It has three challenges to unlock the evidence devices on Live Forensics. They are (i) USB Debugging Mode (ii) Authentication Access with RSA key (iii) Root Access.

USB Debugging Mode – If this mode is enabled, it can run ADB commands and detect the android devices. ADB is a command line software interface between an android device and a host computer. If this mode is disabled, it may be difficult to continue next step.

Authentication Access with RSA key - In the jelly bean (v4.2.2) update, designers introduced the new security policy on ADB called “secure USB debugging”. So, even if USB debugging mode is enabled, it need authentication access to run ADB command. When the device is connected to a host, the device sends a 20-byte random message to the host. The host encrypts the message’s SHA1 with RSA signature using its private key and sends this encrypted signature back. If investigator will get related laptop/PC that not only already accepted RSA key of evidence device but also didn’t revoke, it can run next step for root check.

Root Access – It is a very important thing in forensics workflow because some process of android forensics need root access. But android restricts the user levels and modify system processes. If evidence devices are not already rooted, it may be difficult to unlock the evidence devices on Live Forensics. Because most of the techniques of rooting process need to reboot the device and it will be lost volatile data on memory. In proposed framework, it collected the different safe root tools to provide for rooting process depend on each device.

Lock Screen Mode on Static Forensics: In static process, it has more chance than live process to unlock the evidence devices. Because it doesn’t need to keep volatile data. So, it can install custom recovery via fastboot mode or downloading mode depend on device brands. And then unlock script file update on custom recovery mode for unlock evidence devices.

Extract volatile data on Android Device: It needs to check and record current process and status before memory imaging. (i) Notification Bar – if airplane mode is on, back off in communication shield and make a note which services is on or off. They are Mobile Data, Wireless, Bluetooth, GPS, Alarm and etc. Otherwise, it can check after memory imaging. (ii) Instant Message – social media application (online) and mobile SMS (offline) (iii) Recent Applications and Files (example. Long press Home Key).
Memory imaging is very important process of Live Forensics. Memory stores and keeps a lot of raw volatile data. In Dalvik Virtual Machine, [12] Eclipse (Memory Analyzer Tool) and [13] Android Studio tool (Memory Monitoring) can capture memory process on each application with HPROF binary heap dumps file. Many researchers already proposed [14] LiME (Linux Memory Extractor) and [15] Volatility tool for memory imaging. In this proposed framework, it used these freeware or open source tools for extract volatile data on memory, besides Linux command-line. Example, “*cat /dev/mem > /storage/sdcard0/mem.txt*” – it produces a lot of raw data from memory to sdcard with text file. Additionally, “*cat /dev/log/ “__’*””, “*cat /dev/kmsg*”, adb logcat, adb shell dumpsys are also produce a lot of information.

Android Storage Imaging: Android has two types of partition that are Physical partition and Logical partition. Android forensics need to investigate both partitions and ext-sd card partition. Physical partition imaging, it creates entire copy of internal memory and it includes unallocated space. Logical partition has various types of file system partitions. They are boot, system, recovery, data, cache, misc and etc. This framework used “*dd*” command for android storage imaging. This command is a Linux command-line utility and it can create bit-by-bit images both of Physical and Logical partition. Otherwise, android internal storage divided various block types and *dd* command can create an image of each block.

VI. EXPERIMENTAL RESULTS

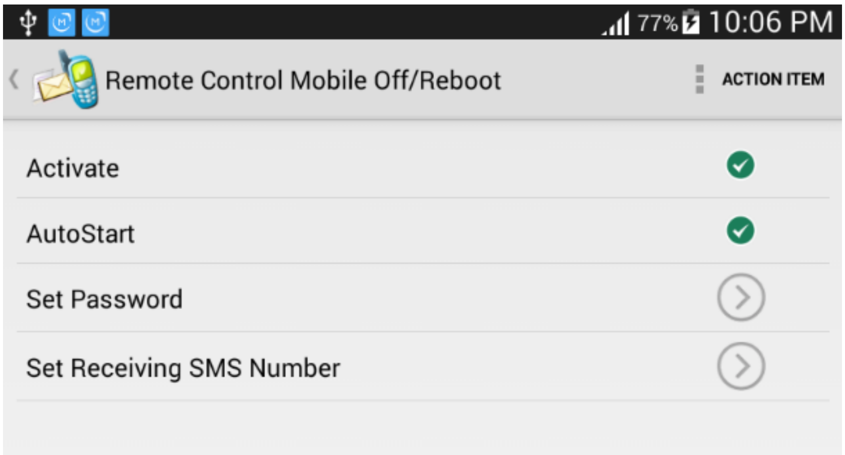
In this paper, we already tested on two android versions Jelly Bean 4.1.1 (no RSA key feature) and KitKat 4.4.2. Samsung and Huawei are very popular in smartphone market of developing countries. Therefore, Samsung and Huawei were chosen as the test case and profile detail shown in Table 1.

Table 1. Test devices profile

Property	**Samsung**	**Huawei**
Name	Galaxy Tab 4	Ascend
Model	SM-T231	G510-0200
Android Version	4.4.2	4.1.1
Build Number	KOT49H.T231 XXU0AOL2	G510-0200V100R 001 C00B173

In this framework, we have tested various tools and commands on 64-bits OS based Ubuntu 16.04 LTS and Window 10. The following are some practical processes on the android devices with power on state and lock screen mode (already root):

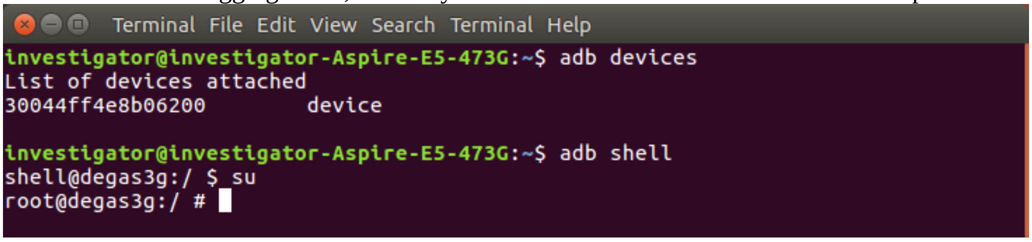
Testing 1: Why need to disconnect the communication of Android smartphone?
Criminals may be installed some application to control the device. For example, Power off Root Req_v1.0 application can restart or power off to the device via phone SMS. So, volatile data can be lost when investigator collects the evidence devices. So, it need to airplane mode ON before backup and data collection.



Activate ✓
AutoStart ✓
Set Password
Set Receiving SMS Number

Fig 5 Remote control application

Testing 2: Check USB debugging mode, RSA key authentication and Root access for unlock process.



```
investigator@investigator-Aspire-E5-473G:~$ adb devices
List of devices attached
30044ff4e8b06200        device

investigator@investigator-Aspire-E5-473G:~$ adb shell
shell@degas3g:/ $ su
root@degas3g:/ #
```

Fig 6 Fully access state (USB debugging, RSA key and Root)

If investigator got fully access to evidence device as Fig 6, ADB commands can remove or copy/paste key of screen lock for unlock the device. For examples, *adb pull*, *adb push*, *adb remove* to key directory: (*/data/system/ gesture.key* for pattern lock and */data/system/password.key* for pin or password. But KitKat to latest versions have challenges to get RSA authentication access on android devices. It may be losing live forensics process to unlock evidence devices.

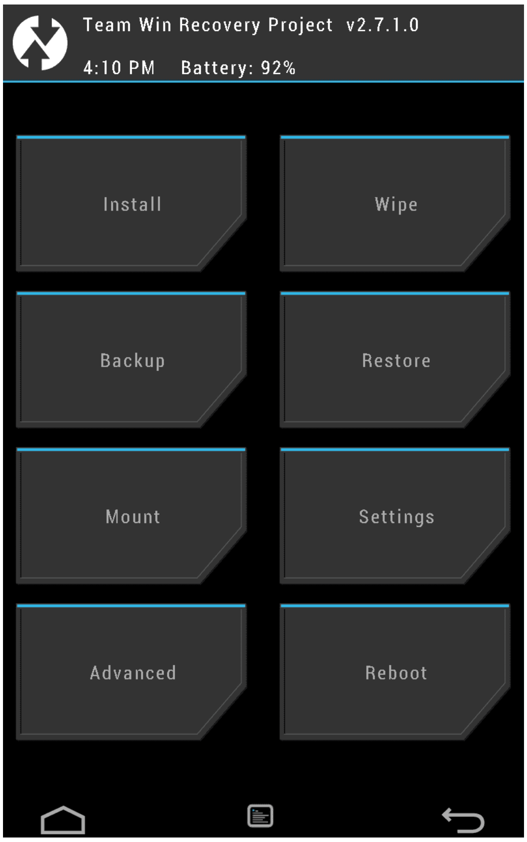
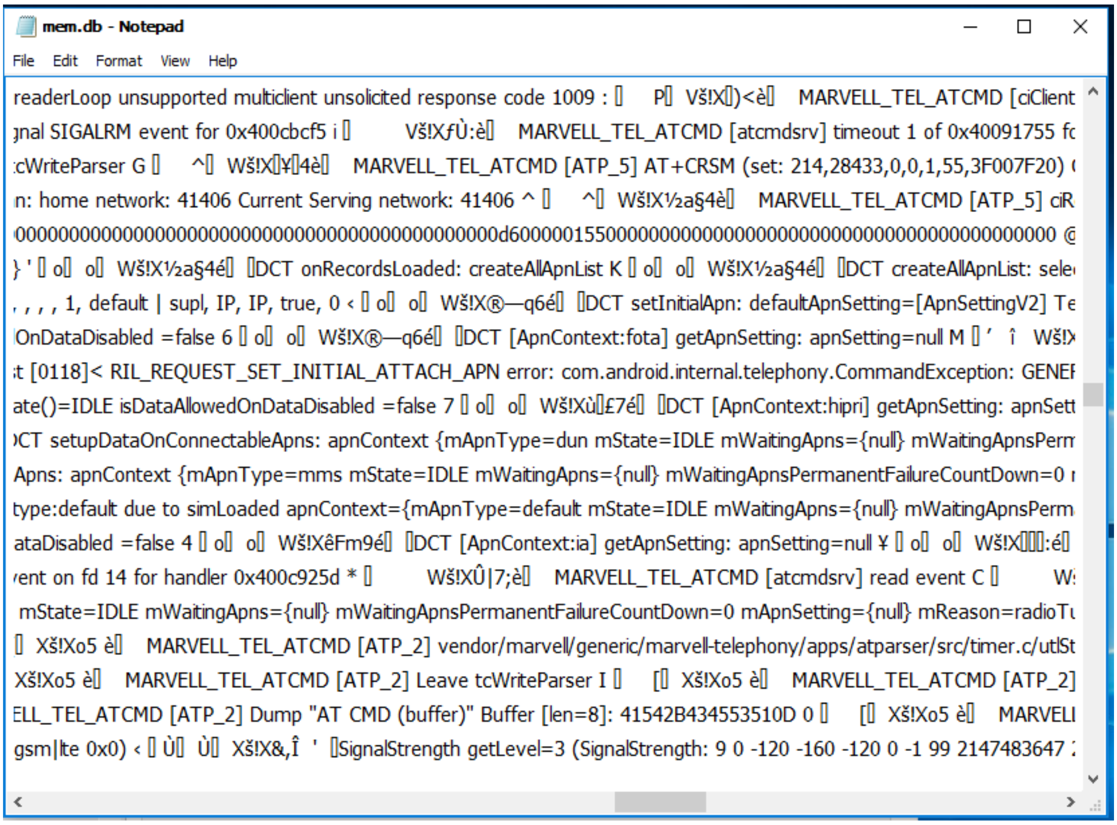


Fig 7 TWRP Custom recovery on Samsung Galaxy Tab 4

If investigator cannot get fully access to evidence devices, it is needed to install custom recovery for unlock. Custom recovery has useful features and it can unlock or root process evidence devices by update zip files as Fig 7. But it is also needed to reboot and downloading mode during installation process and it may be losing volatile data.

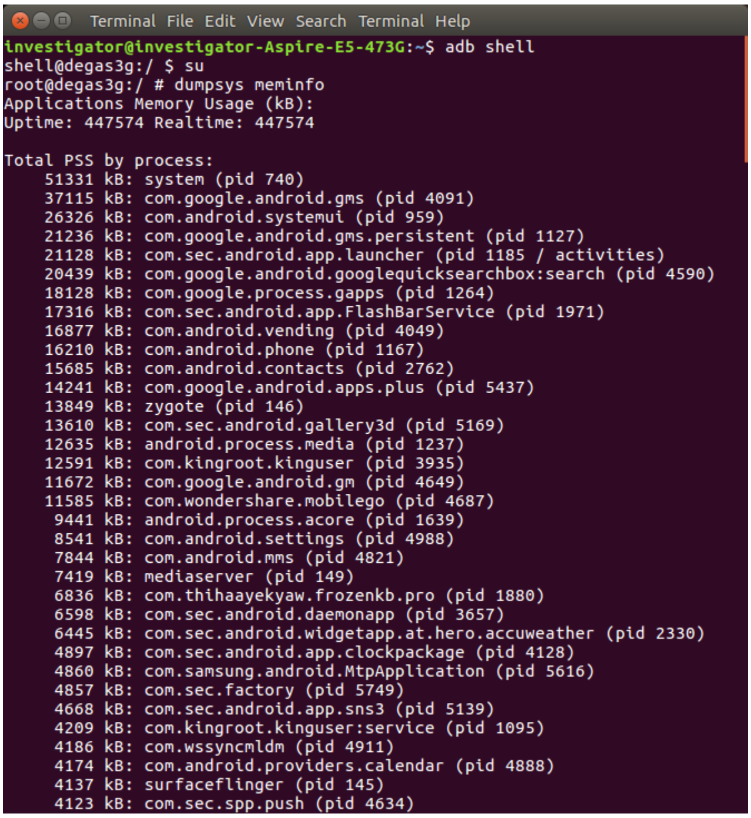
Testing 3: It can be recorded manually or adb commands for current process and status. And then it also needs to backup by adb command without root access. For example, backup command is *adb backup -apk -shared -all -f /backup/backup.ab* and restore command is adb restore /backup/backup.ab.

Testing 4: In extract volatile data on memory, it can use Eclipse, Android Studio, LiME and *dd* commands. Additionally, following some adb commands are already tested for extract volatile data and logs: *cat /dev/mem > /storage/sdcard0/ mem.txt*, *adb dumpsys* and *adb dumpsys meminfo*, *adb logcat*. These commands can get rich amount of raw volatile data.



```
readerLoop unsupported multiclient unsolicited response code 1009 : ▯    P▯  Vš!X▯)<è▯   MARVELL_TEL_ATCMD [ciClient
ˌnal SIGALRM event for 0x400cbcf5 i ▯        Vš!XƒÙ:è▯   MARVELL_TEL_ATCMD [atcmdsrv] timeout 1 of 0x40091755 fc
:cWriteParser G ▯    ^▯  Wš!X▯¥▯4è▯   MARVELL_TEL_ATCMD [ATP_5] AT+CRSM (set: 214,28433,0,0,1,55,3F007F20) (
n: home network: 41406 Current Serving network: 41406 ^ ▯    ^▯  Wš!X½a§4è▯   MARVELL_TEL_ATCMD [ATP_5] ciR
000000000000000000000000000000000000000000d6000015500000000000000000000000000000000000000000 @
} ' ▯ o▯  o▯  Wš!X½a§4é▯  ▯DCT onRecordsLoaded: createAllApnList K ▯ o▯  o▯  Wš!X½a§4é▯  ▯DCT createAllApnList: sele
, , , , 1, default | supl, IP, IP, true, 0 ‹ ▯ o▯  o▯  Wš!X®—q6é▯  ▯DCT setInitialApn: defaultApnSetting=[ApnSettingV2] Te
OnDataDisabled =false 6 ▯ o▯  o▯  Wš!X®—q6é▯  ▯DCT [ApnContext:fota] getApnSetting: apnSetting=null M ▯ ʼ  î   Wš!X
:t [0118]< RIL_REQUEST_SET_INITIAL_ATTACH_APN error: com.android.internal.telephony.CommandException: GENEF
ate()=IDLE isDataAllowedOnDataDisabled =false 7 ▯ o▯  o▯  Wš!Xù▯£7é▯  ▯DCT [ApnContext:hipri] getApnSetting: apnSett
)CT setupDataOnConnectableApns: apnContext {mApnType=dun mState=IDLE mWaitingApns={null} mWaitingApnsPerm
Apns: apnContext {mApnType=mms mState=IDLE mWaitingApns={null} mWaitingApnsPermanentFailureCountDown=0 r
type:default due to simLoaded apnContext={mApnType=default mState=IDLE mWaitingApns={null} mWaitingApnsPerm
ataDisabled =false 4 ▯ o▯  o▯  Wš!XêFm9é▯  ▯DCT [ApnContext:ia] getApnSetting: apnSetting=null ¥ ▯ o▯  o▯  Wš!X▯▯▯:é▯
vent on fd 14 for handler 0x400c925d * ▯        Wš!XÛ|7;è▯   MARVELL_TEL_ATCMD [atcmdsrv] read event C ▯        W:
 mState=IDLE mWaitingApns={null} mWaitingApnsPermanentFailureCountDown=0 mApnSetting={null} mReason=radioTu
▯  Xš!Xo5 è▯   MARVELL_TEL_ATCMD [ATP_2] vendor/marvell/generic/marvell-telephony/apps/atparser/src/timer.c/utlSt
Xš!Xo5 è▯   MARVELL_TEL_ATCMD [ATP_2] Leave tcWriteParser I ▯    [▯  Xš!Xo5 è▯   MARVELL_TEL_ATCMD [ATP_2]
ELL_TEL_ATCMD [ATP_2] Dump "AT CMD (buffer)" Buffer [len=8]: 41542B434553510D 0 ▯    [▯  Xš!Xo5 è▯   MARVELI
gsm|lte 0x0) ‹ ▯ Ù▯  Ù▯  Xš!X&,Î  '  ▯SignalStrength getLevel=3 (SignalStrength: 9 0 -120 -160 -120 0 -1 99 2147483647 :
```

Fig 8 Raw data in output text file (adb command - cat)

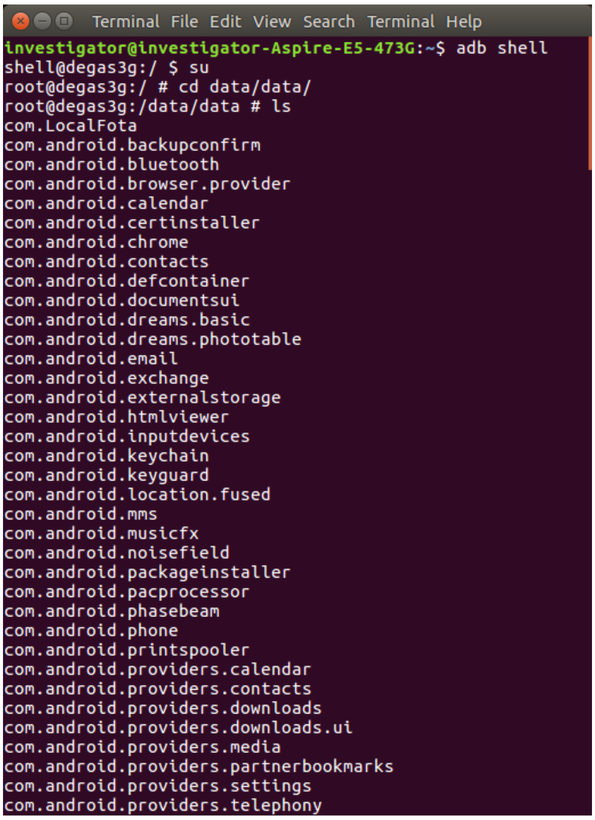


```
investigator@investigator-Aspire-E5-473G:~$ adb shell
shell@degas3g:/ $ su
root@degas3g:/ # dumpsys meminfo
Applications Memory Usage (kB):
Uptime: 447574 Realtime: 447574

Total PSS by process:
    51331 kB: system (pid 740)
    37115 kB: com.google.android.gms (pid 4091)
    26326 kB: com.android.systemui (pid 959)
    21236 kB: com.google.android.gms.persistent (pid 1127)
    21128 kB: com.sec.android.app.launcher (pid 1185 / activities)
    20439 kB: com.google.android.googlequicksearchbox:search (pid 4590)
    18128 kB: com.google.process.gapps (pid 1264)
    17316 kB: com.sec.android.app.FlashBarService (pid 1971)
    16877 kB: com.android.vending (pid 4049)
    16210 kB: com.android.phone (pid 1167)
    15685 kB: com.android.contacts (pid 2762)
    14241 kB: com.google.android.apps.plus (pid 5437)
    13849 kB: zygote (pid 146)
    13610 kB: com.sec.android.gallery3d (pid 5169)
    12635 kB: android.process.media (pid 1237)
    12591 kB: com.kingroot.kinguser (pid 3935)
    11672 kB: com.google.android.gm (pid 4649)
    11585 kB: com.wondershare.mobilego (pid 4687)
     9441 kB: android.process.acore (pid 1639)
     8541 kB: com.android.settings (pid 4988)
     7844 kB: com.android.mms (pid 4821)
     7419 kB: mediaserver (pid 149)
     6836 kB: com.thihaayekyaw.frozenkb.pro (pid 1880)
     6598 kB: com.sec.android.daemonapp (pid 3657)
     6445 kB: com.sec.android.widgetapp.at.hero.accuweather (pid 2330)
     4897 kB: com.sec.android.app.clockpackage (pid 4128)
     4860 kB: com.samsung.android.MtpApplication (pid 5616)
     4857 kB: com.sec.factory (pid 5749)
     4668 kB: com.sec.android.app.sns3 (pid 5139)
     4209 kB: com.kingroot.kinguser:service (pid 1095)
     4186 kB: com.wssyncmldm (pid 4911)
     4174 kB: com.android.providers.calendar (pid 4888)
     4137 kB: surfaceflinger (pid 145)
     4123 kB: com.sec.spp.push (pid 4634)
```

Fig 9 Memory information (dumpsys meminfo)

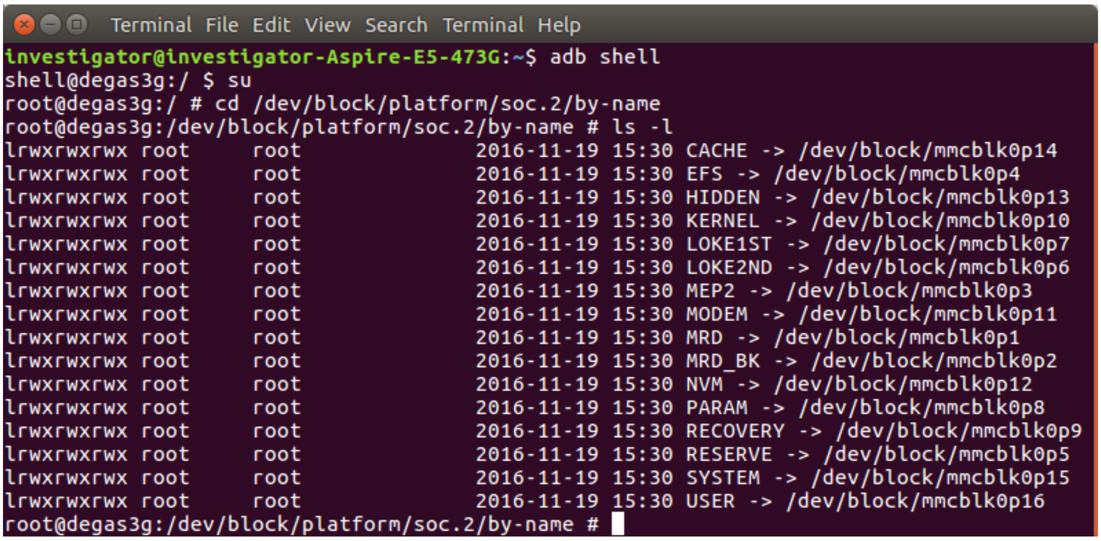
Testing 5: To investigate the related application, it need to extract the data of application. Directory is */data/data/"app_name"* and most of the evidences data are stored in shared_prefs, cache and databases files.



```
investigator@investigator-Aspire-E5-473G:~$ adb shell
shell@degas3g:/ $ su
root@degas3g:/ # cd data/data/
root@degas3g:/data/data # ls
com.LocalFota
com.android.backupconfirm
com.android.bluetooth
com.android.browser.provider
com.android.calendar
com.android.certinstaller
com.android.chrome
com.android.contacts
com.android.defcontainer
com.android.documentsui
com.android.dreams.basic
com.android.dreams.phototable
com.android.email
com.android.exchange
com.android.externalstorage
com.android.htmlviewer
com.android.inputdevices
com.android.keychain
com.android.keyguard
com.android.location.fused
com.android.mms
com.android.musicfx
com.android.noisefield
com.android.packageinstaller
com.android.pacprocessor
com.android.phasebeam
com.android.phone
com.android.printspooler
com.android.providers.calendar
com.android.providers.contacts
com.android.providers.downloads
com.android.providers.downloads.ui
com.android.providers.media
com.android.providers.partnerbookmarks
com.android.providers.settings
com.android.providers.telephony
```

Fig 10 Data directories of Applications

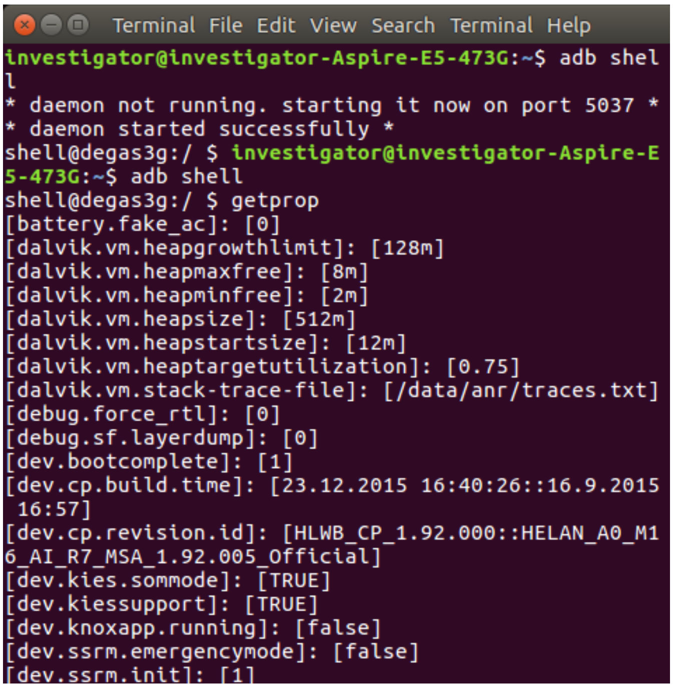
Testing 6: Imaging the storage devices, *dd* command can create an image of partitions and blocks. Example of *dd* command: "*dd if=/dev/block/ mmcblk0 of=/sdcard/blk0.img bs=4096*" (*if*: the directory of input file, *of*: the directory of output file, *bs*: block size, defaults to 512 bytes). Additionally, netcat command (*nc*) can create an image to hosted computer without SD card. And then mount to dd image file: *sudo mount blk0.img /mnt* (or) [15] FTK imager tool.



```
investigator@investigator-Aspire-E5-473G:~$ adb shell
shell@degas3g:/ $ su
root@degas3g:/ # cd /dev/block/platform/soc.2/by-name
root@degas3g:/dev/block/platform/soc.2/by-name # ls -l
lrwxrwxrwx root     root              2016-11-19 15:30 CACHE -> /dev/block/mmcblk0p14
lrwxrwxrwx root     root              2016-11-19 15:30 EFS -> /dev/block/mmcblk0p4
lrwxrwxrwx root     root              2016-11-19 15:30 HIDDEN -> /dev/block/mmcblk0p13
lrwxrwxrwx root     root              2016-11-19 15:30 KERNEL -> /dev/block/mmcblk0p10
lrwxrwxrwx root     root              2016-11-19 15:30 LOKE1ST -> /dev/block/mmcblk0p7
lrwxrwxrwx root     root              2016-11-19 15:30 LOKE2ND -> /dev/block/mmcblk0p6
lrwxrwxrwx root     root              2016-11-19 15:30 MEP2 -> /dev/block/mmcblk0p3
lrwxrwxrwx root     root              2016-11-19 15:30 MODEM -> /dev/block/mmcblk0p11
lrwxrwxrwx root     root              2016-11-19 15:30 MRD -> /dev/block/mmcblk0p1
lrwxrwxrwx root     root              2016-11-19 15:30 MRD_BK -> /dev/block/mmcblk0p2
lrwxrwxrwx root     root              2016-11-19 15:30 NVM -> /dev/block/mmcblk0p12
lrwxrwxrwx root     root              2016-11-19 15:30 PARAM -> /dev/block/mmcblk0p8
lrwxrwxrwx root     root              2016-11-19 15:30 RECOVERY -> /dev/block/mmcblk0p9
lrwxrwxrwx root     root              2016-11-19 15:30 RESERVE -> /dev/block/mmcblk0p5
lrwxrwxrwx root     root              2016-11-19 15:30 SYSTEM -> /dev/block/mmcblk0p15
lrwxrwxrwx root     root              2016-11-19 15:30 USER -> /dev/block/mmcblk0p16
root@degas3g:/dev/block/platform/soc.2/by-name # █
```

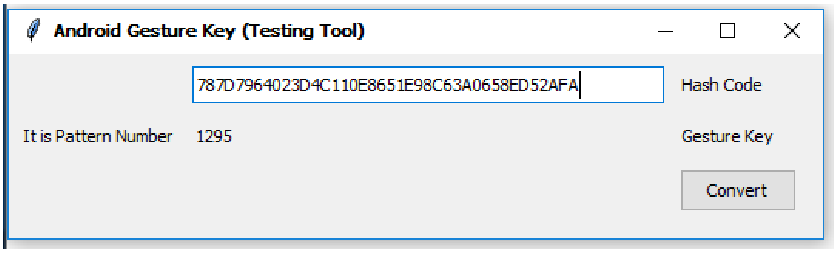
Fig 11 Detail blocks for storage imaging

Testing 7: Examination and Analysis, it contains device profiles, installed app, current process, contacts, message and etc. The *getprop* command is useful for that. And this framework already tested some freeware tools are [16] AFLogical OSE [17] Hex Editor tool, [18] SQLite DB Browser and Proposed tool for decode and analysis. Proposed tool in framework, it provides to decode the pattern lock for unlock devices. This tool is very easy to use and understand.



```
investigator@investigator-Aspire-E5-473G:~$ adb shel
l
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
shell@degas3g:/ $ investigator@investigator-Aspire-E
5-473G:~$ adb shell
shell@degas3g:/ $ getprop
[battery.fake_ac]: [0]
[dalvik.vm.heapgrowthlimit]: [128m]
[dalvik.vm.heapmaxfree]: [8m]
[dalvik.vm.heapminfree]: [2m]
[dalvik.vm.heapsize]: [512m]
[dalvik.vm.heapstartsize]: [12m]
[dalvik.vm.heaptargetutilization]: [0.75]
[dalvik.vm.stack-trace-file]: [/data/anr/traces.txt]
[debug.force_rtl]: [0]
[debug.sf.layerdump]: [0]
[dev.bootcomplete]: [1]
[dev.cp.build.time]: [23.12.2015 16:40:26::16.9.2015
 16:57]
[dev.cp.revision.id]: [HLWB_CP_1.92.000::HELAN_A0_M1
6_AI_R7_MSA_1.92.005_Official]
[dev.kies.sommode]: [TRUE]
[dev.kiessupport]: [TRUE]
[dev.knoxapp.running]: [false]
[dev.ssrm.emergencymode]: [false]
[dev.ssrm.init]: [1]
```

Fig 12 Device properties (getprop command)



787D7964023D4C110E8651E98C63A0658ED52AFA Hash Code

It is Pattern Number 1295 Gesture Key

Fig 13 Proposed tool (decode the pattern lock)

VII.CONCLUSION

Forensics organizations in developing countries, they always need to improve their process, framework and tools. Especially, our country (Myanmar) is very new experience for android forensics in cyber-crime investigation. This proposed work is application for Myanmar that process flow provides applicable guidelines and analysis framework can extensible when a new tool or module develops in it. And it could be plugged in to the framework easily for forensics organizations. But this proposed framework cannot analyze some specific fields as Malware Analysis. However, we are preparing to deeply analysis for some android forensics fields based on proposed framework for future work:

- Anti-forensics
- Steganography
- Ransomware
- Social Media Application

Now above specific fields are very hot topics in research fields. And more tools are going to develop for further forensics analysis. So, our research work is definitely applicable in cyber-crime investigation.

REFERENCES

[1]. Karen Kent, Suzanne Chevalier, Tim Grance and Hung Dang, *Guide to integrating forensic techniques into incident response*, (NIST Special Publication 800-86, August 2006).

[2]. Rick Ayers, Sam Brothers and Wayne Jansen, *Guidelines on mobile device forensics*, (NIST Special Publication 800-101, May 2014).

[3]. 2016 Android Operating System [online] Available: https://en.wikipedia.org/wiki/Android_(operating_system)#Software_stack.

[4]. Haroon Q Raja, *Android Partitions Explained: boot, system, recovery, data, cache & misc*, http://www.addictivetips.com/mobile/androidpartitions-explained-boot-system-recovery-data-cache-misc/, May 19, 2011.

[5]. ISAK MRKAIĆ, *Android forensic using some open source tools*, The Eighth International Conference on Business Information Security (BISEC-2016), Belgrade, Serbia, , 15th October 2016.

[6]. Archit Goel, Anurag Tyagi and Ankit Agarwal, *Smartphone Forensic Investigation Process Model*, International Journal of Computer Science & Security (IJCSS), (Northern India Engineering College, 2012).

[7]. Indeewari U. Akarawita, Amila B. Perera and Ajantha Atukorale, *ANDROPHSY – Forensic Framework for Android*, International Conference on Advances in ICT for Emerging Regions (ICTer): 250-258, (University of Colombo School of Computing, Sri Lanka, August 2015).

[8]. Abdalazim Abdallah Mohammed Alamin and Dr. Amin Babiker A/Nabi Mustafa, *Implementing Digital Forensic Framework for Android Smart Phones, International Journal of Engineering, Applied and Management Sciences Paradigms, Vol. 21*, Issue 01, Al-Neelain University, Sudan, February 2015.

[9]. Muhammad Faheem, N.-A. Le-Khac and Tahar Kechadi, *Smartphone Forensic Analysis: A Case Study for Obtaining Root Access of an Android Samsung S3 Device and Analyse the Image without an Expensive Commercial Tool*, *Journal of Information Security, 5*, 83-90, University College Dublin, Dublin, Ireland, July 2015.

[10]. Ben Martini, Quang Do and Kim-Kwang Raymond Choo, *Conceptual Evidence Collection and Analysis Methodology for Android Devices*, http://dx.doi.org/10.1016/B978-0-12-801595-7.00014-8, Information Assurance Research Group, (University of South Australia, 2015, pp. 285-307, Syngress, an Imprint of Elsevier).

[11]. Andri P. Heriyanto, *Procedures and Tools For Acquisition And Analysis Of Volatile Memory On Android Smartphones,* Australian Digital Forensics Conference, (Edith Cowan University, December 2013.)

[12]. http://www.eclipse.org/mat/

[13]. https://developer.android.com/studio/profile/am-memory.html

[14]. http://www.volatilityfoundation.org/

[15]. https://accessdata.com/product-download/digital-forensics/ftk-imager-version-3.2.0

[16]. https://github.com/nowsecure/android-forensics/ downloads

[17]. http://www.hhdsoftware.com/Downloads/free-he x-editor

[18]. http://sqlitebrowser.org/

Biographies and Photographs

Name – Mr. Naing Linn Htun
Age – 30
Material Status - Single
Graduated – M.I.Sc, Computer University, Maubin
Current Education –
Ph.D. Research Candidate (10th Batch), Cyber Security Research Lab, University of Computer Studies, Yangon (UCSY)
Nationality – Myanmar

Working Experiences
– Mobile Services Technician, Lucky Century Company Limited (IT services and shop) (2 years and 6 months)
- IT specialist (Computer, Mobile, Hotel Management Software (Optima), HP Graphic Solution Printer), Lucky Bird Group of Companies (IT services and shop) (2 years)

Name – Dr. Mie Mie Su Thwin
Age – 45
Material Status – Single
Graduated – Ph.D Computer Science, University of Computer Studies, Yangon
Working Experiences – Associate Professor (20 years), mmCERT (6 years)