A Study of User Data Integrity During Acquisition of Android Devices

By

Namheun Son, Yunho Lee, Dohyun Kim,
Joshua I. James, Sangjin Lee and Kyungho Lee

Presented At

The Digital Forensic Research Conference

DFRWS 2013 USA Monterey, CA (Aug 4th - 7th)

DFRWS is dedicated to the sharing of knowledge and ideas about digital forensics research. Ever since it organized the first open workshop devoted to digital forensics in 2001, DFRWS continues to bring academics and practitioners together in an informal environment. As a non-profit, volunteer organization, DFRWS sponsors technical working groups, annual conferences and challenges to help drive the direction of research and development.

http:/dfrws.org

DFRWS 2013

A Study of User Data Integrity During Acquisition of Android Devices

Namheun Son, Yunho Lee, Dohyun Kim, Joshua I. James, Sangjin Lee, and Kyungho Lee

2013-08-05

Dohyun Kim

exdus84@gmail.com

Digital Forensic Research Center

Center for Information Security Technologies, Korea University

Agenda

1. Introduction
2. Related Work
3. Background
4. Process of User Data Acquisition
5. Android Extractor
6. Experiment using Android Extractor
7. Demonstration
8. Conclusion

Introduction

- **Android**
- **Booting Mode**

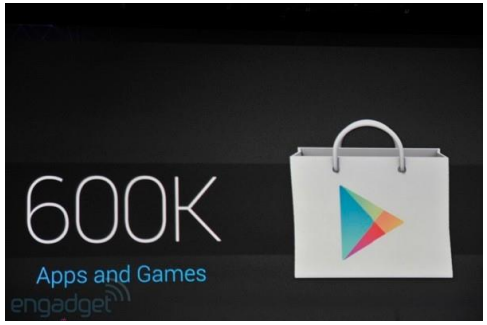
Android Forensics

- **Android OS Market Share (Q3, 2013)**

Top Six Smartphone Mobile Operating Systems, Shipments, and [illegible]inary)
(Units in Millions)

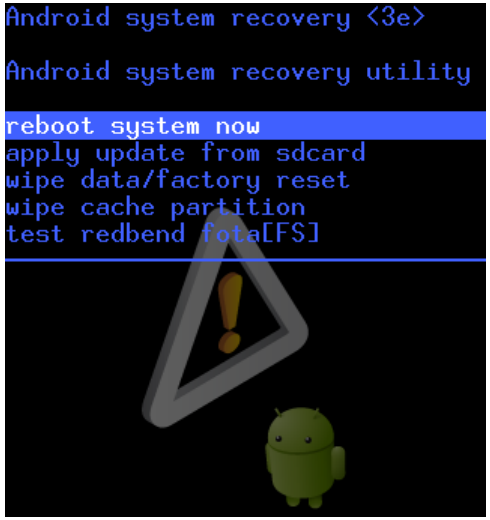
Operating System	3Q12 Shipment Volumes	3Q
Android	1	
iOS		
BlackBerry		
Symbian		
Windows Phone 7/ Windows Mobile		
Linux		
Others		
Totals		

- **600,000 Apps & inst**
  - 50 app installs per Android d



Booting Mode

Recovery Mode



- **A sort of standard booting mode**
  - Holding keys during boot process
    - Ex) volume key + power key + …
- **Feature**
  - wiping partitions, install an system application, etc…
  - Not mount userdata partition
- **We use recovery mode for acquisition user data**
  - DFRWS 2011 : “Toward a general collection methodology for Android devices”, Vidas

Related Work

- Logical Method of Data Acquisition
- Physical Method of Data Acquisition
- Commercial Tools

Logical Method of Data Acquisition

Content Provider

- **Data sharing interface for application level**
  - Supporting data between different applications
- **Android use a Sandbox mechanism for security**
  - An application can not approach another application data
    - Have to know application's URI for access application's data
    - Google make public android default application's URI
      - Call history, Contacts, SMS/MMS, etc..
- **Have to install an application contain the content provider**
  - For acquisition other application data
  - User data area could be altered

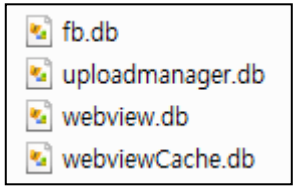
Logical Method of Data Acquisition

ADB (Android debug Bridge) protocol

- **File unit, partition**

```
# ls -l
ls -l
drwxr-xr-x app_0    app_0          2010-07-19 12:14 com.sec.mms
drwxr-xr-x app_1    app_1          2010-07-19 12:12 com.sec.android.app.appinstaller
drwxr-xr-x app_131  app_131        2011-01-02 16:00 com.sec.android.app.dlna
drwxr-xr-x app_2    app_2          2010-07-19 12:12 com.sec.android.app.drmua
drwxr-xr-x app_3    app_3          2010-07-19 12:12 com.svox.pico
drwxr-xr-x radio    radio          2010-07-19 12:12 com.android.stk
```

```
E:₩>adb pull /data/data/com.facebook.katana/databases .₩facebook_file
pull: building file list...
pull: /data/data/com.facebook.katana/databases/webviewCache.db -> .₩facebook_file/webviewCache.db
pull: /data/data/com.facebook.katana/databases/webview.db -> .₩facebook_file/webview.db
pull: /data/data/com.facebook.katana/databases/uploadmanager.db -> .₩facebook_file/uploadmanager.db
pull: /data/data/com.facebook.katana/databases/fb.db -> .₩facebook_file/fb.db
4 files pulled. 0 files skipped.
1375 KB/s (277504 bytes in 0.197s)
```



Logical Method of Data Acquisition

Rooting

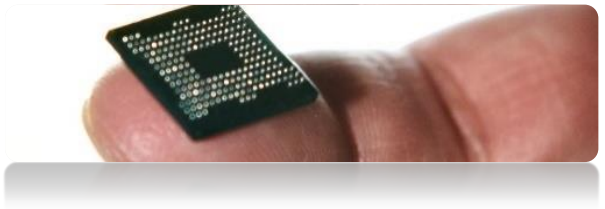
- **Temporary Rooting**
  - Using by exploit
    - psneuter, regeagainstthecase, zergRush, and so on…
- **Full Rooting**
  - Using by custom booting kernel image
- **Possible to get all data from android device**
  - Imaging partitions, copying files
    - Using ADB protocol and DD binaries

    ➔ Faster than JTAG

Physical Method of Data Acquisition

Chip-off

- **Directly separates flash memory**
  - From embedded device board



- **Possible to get damaged in process of separating Flash memory**
  - Smartphone, Flash memory

- **Necessary to data reconstruction (File System)**
  - Raw data → logical data
  - Ext file system

Physical Method of Data Acquisition

JTAG

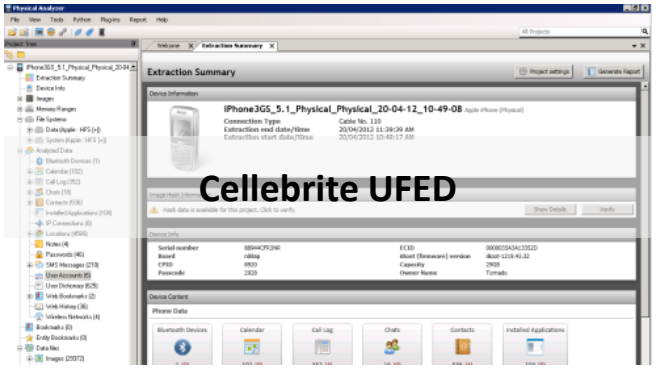
- **Via JTAG debug port**

- **Can acquisition all flash memory data**
  - 0x00 offset – 0xEnd offset

- **But..!**
  - Take so long time to extract data
    - 1GB / 1hour
  - Have to detect JTAG debug port
    - Some android smartphone don’t have a JTAG debug port

Commercial Tools

- **Cellebrite UFED**
  - Use an exploit for rooting
    - To data partition
    - So, data partition can be altered



- **XRY**
  - Data acquisition is possible only when device rooted

Background

- File System of Android Device
- Data Integrity

File System of Android Device

File System

- **YAFFS2 (Yet Another Flash File System2)**
  - Motorola smartphone
- **RFS (Robust File System)**
  - A few SAMSUNG smartphone
    - Galaxy S, Galaxy Tab
- **ExtX (Extended File System X) : Ext3/4**
  - After Gingerbread (Android 2.3) use Ext4
    - 83.2% of Android devices use Gingerbread or above as of Oct.2012

File System of Android Device

Feature of Ext3/4

- **Ext3/4 File System has a journaling function**
  - So, unallocated and journal areas are altered when a partition is mounted
    - Data structure for the ExtX Superblock

Byte Range	Description
12-15	Number of unallocated blocks
16-19	Number of unallocated inodes
24-27	Block Size
44-47	Last mount time
48-51	Last written time
52-53	Current mount count
88-89	Size of each inode structure
208-223	Journal ID
224-227	Journal inode

Data Integrity

Mount as read-only and Turn off the Android Device

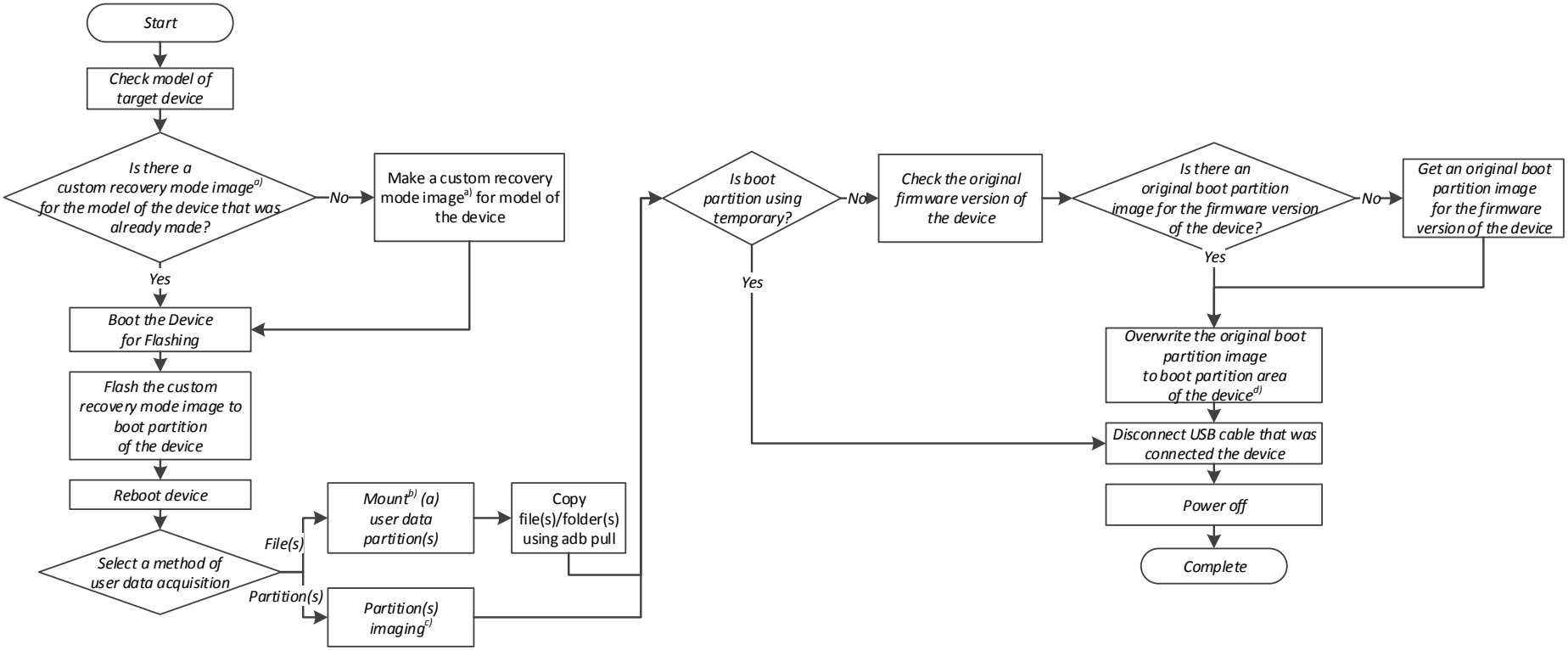
- **Data Integrity is most important for Digital Forensics**

- **Have to prevent alteration**
  - Partition mount as read-only
    - Unallocated area and metadata(superblock) are not altered
  - Turned off the Android device
    - Prevent partition mounting from booting process

Process of User Data Acquisition

- Prepare the Custom Recovery Mode Image (CRMI) (1/6)
- Boot the Device for Flashing (2/6)
- Flash the CRMI to Boot Partition of the Device (3/6)
- User Data Acquisition (4/6)
- Return to Former State (5/6)
- Restoring a Device to Its Original State (6/6)

Process of User Data Acquisition



a) It has to satisfy below conditions.
- Include a busybox binary, enable adb service, has root authority and mount rootf partition as read/write mode

b) The partition(s) has(have) to mount as read only mode.

c) Using adb & dd(cat) or buxybox nanddump(if destination partition is using yaffs2) & busybox netcat

d) Using adb push and dd(cat) or busybox nanddump (if destination partition is using yaffs2)

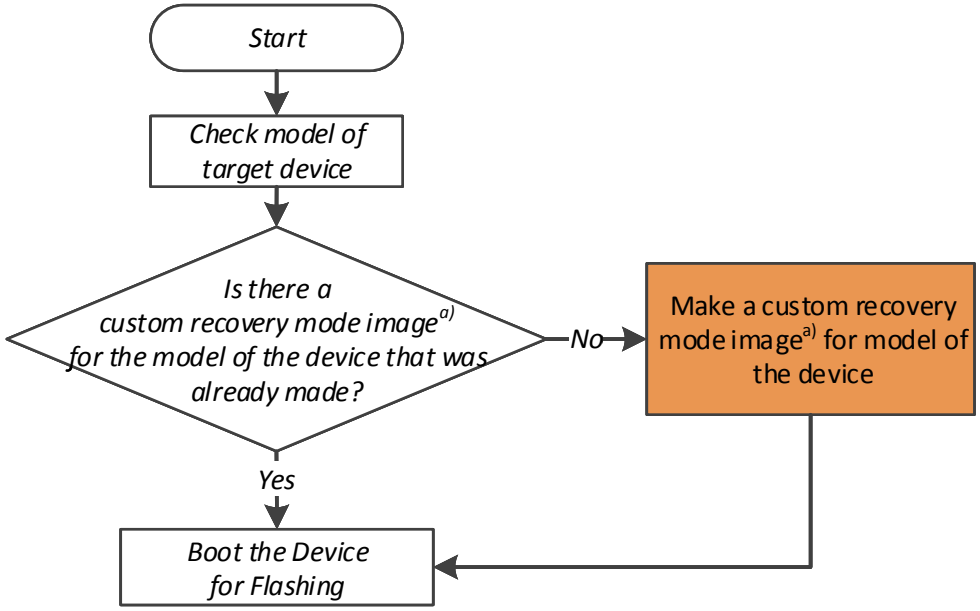
Prepare the Custom Recovery Mode Image (CRMI) (1/6)

1. **Checking the model name of the target device**

2. **Check whether or not the ready Custom Recovery Mode Image (CRMI) for the model**

   ➔ **move next step**

3. **Make a CRMI for model of the device**

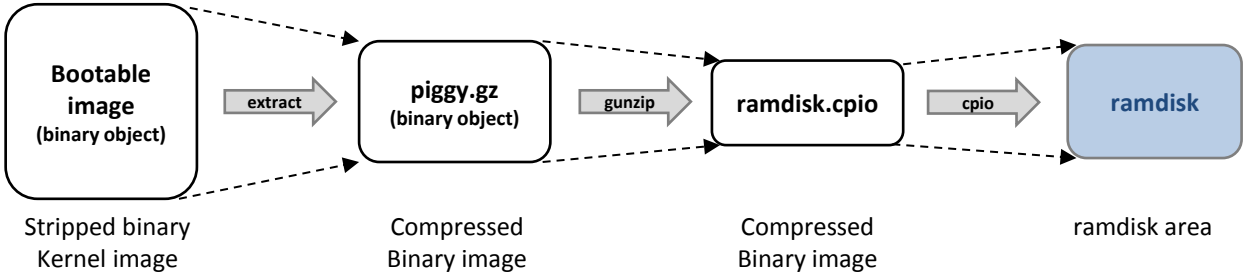


Prepare the Custom Recovery Mode Image (CRMI) (1/6)

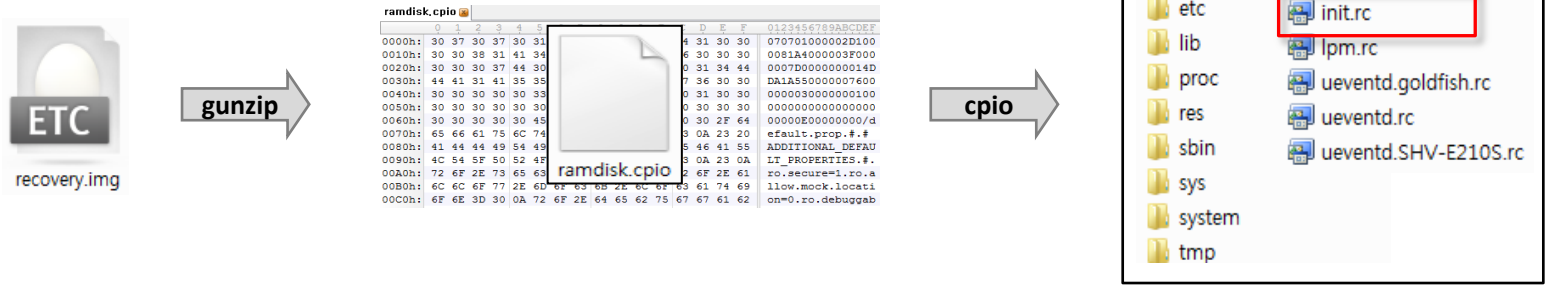
Make a CRMI for Data Integrity

- **Edit a recovery partition ramdisk (modify init.rc, default.prop, adbd file…)**
  - Enable root authority and ADB protocol
  - User data partition unmounts

[Structure of bootable Image of Android]



[Extract ramdisk area]



Prepare the Custom Recovery Mode Image (CRMI) (1/6)

Make a CRMI for Data Integrity

- **Edit a recovery partition ramdisk (modify init.rc, default.prop, adbd file…)**
  - Delete the not related files to booting (resource…)
    - CRMI size should equal the size of the boot partition
      - CRMI should be used for the boot partition
    - Moslty, the size of recovery and boot partition are the same
    - But, some of Android device's recovery partition is bigger than boot partition

Device	Size(Kb)	
	Boot Partition	**Recovery Partition**
Droid (A855)	3584	**4608**
Galaxy S2 (SHW-M250S)	8192	**8192**
Galaxy Nexus (SHW-M420K)	8192	**12224**
Galaxy Note (SHV-E160S)	10240	**10240**
Galaxy S3 (SHV-E210S)	8192	**8192**
Galaxy Note 2 (SHV-E250S)	8192	**8192**
Vega LTE (IM-A800S)	**10240**	**10240**

Boot the Device for Flashing (2/6)

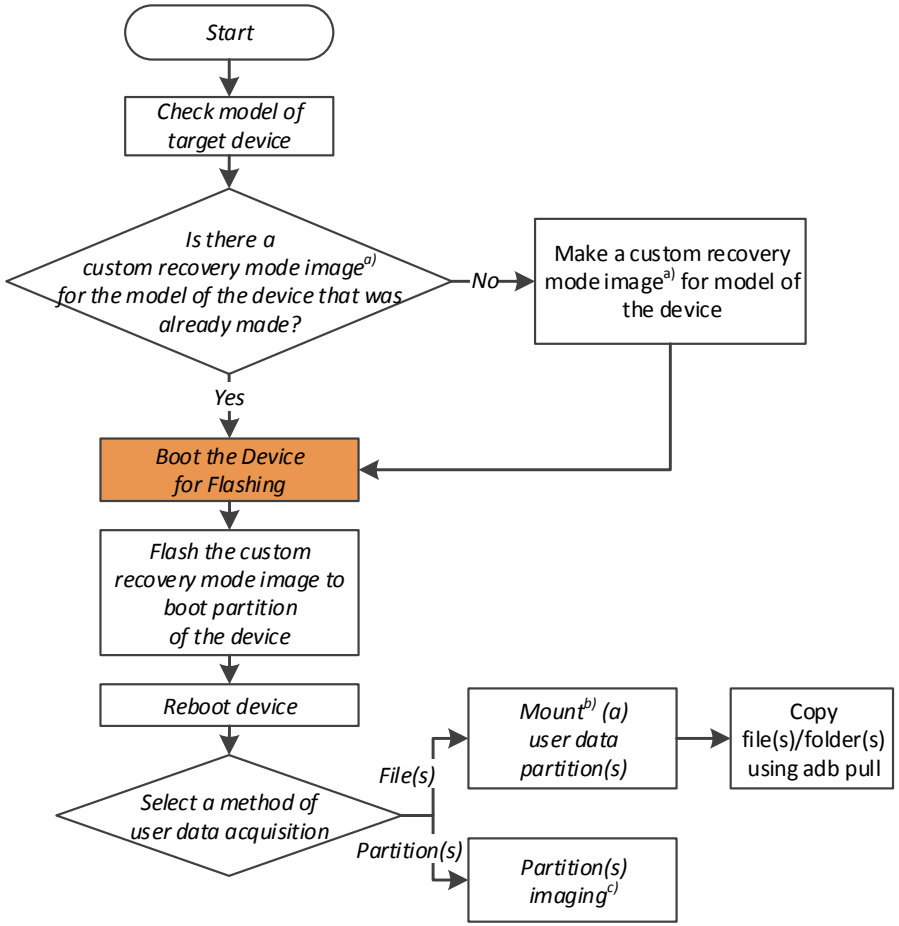
1. **Boot the device in flashing mode for flash the CRMI**

   - Use ADB protocol

     ex) adb reboot download

❖ **Method for entering flashing mode varies for each model**

ex) power key + volume down key + home key



Flash the CRMI to Boot Partition of the Device (3/6)

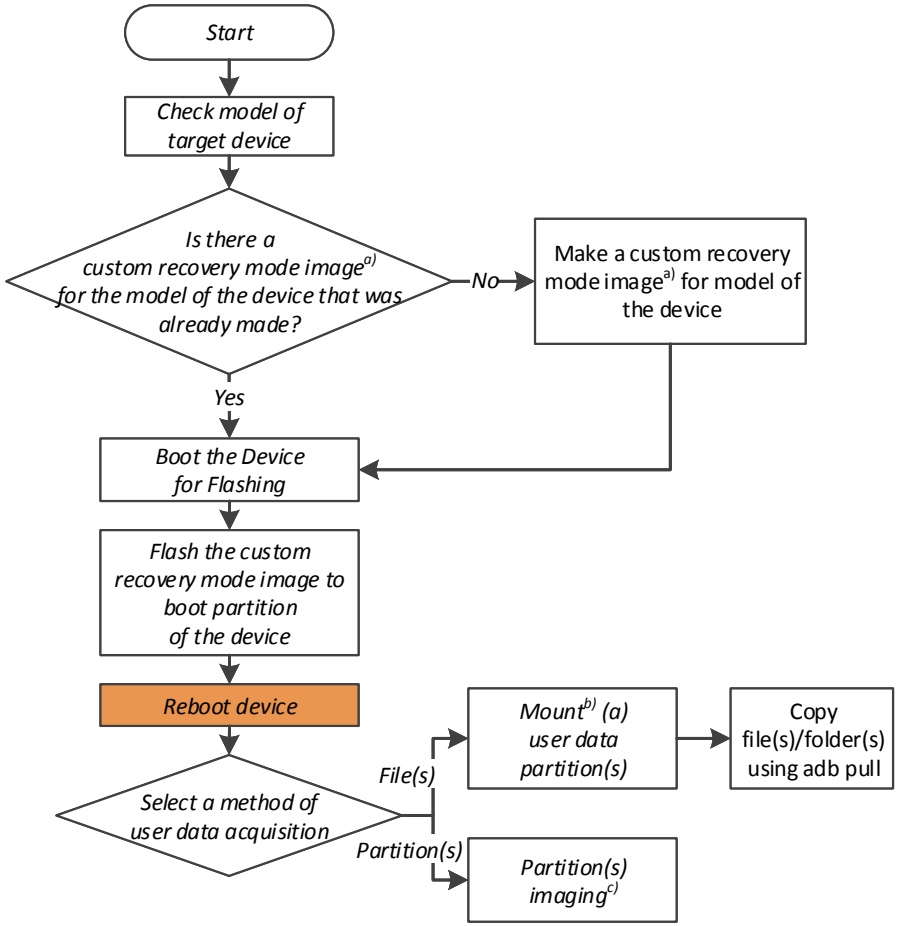
1. **Flash the CRMI to boot partition**

   ➔ Use Odin program

2. **Reboot device**

   ➔ Device is booted recovery mode using CRMI after flashing is finished

   ➔ Acquire root authority



User Data Acquisition (4/6)

1. **Select a method of user data acquisition**

   ➔ Partitions / Files

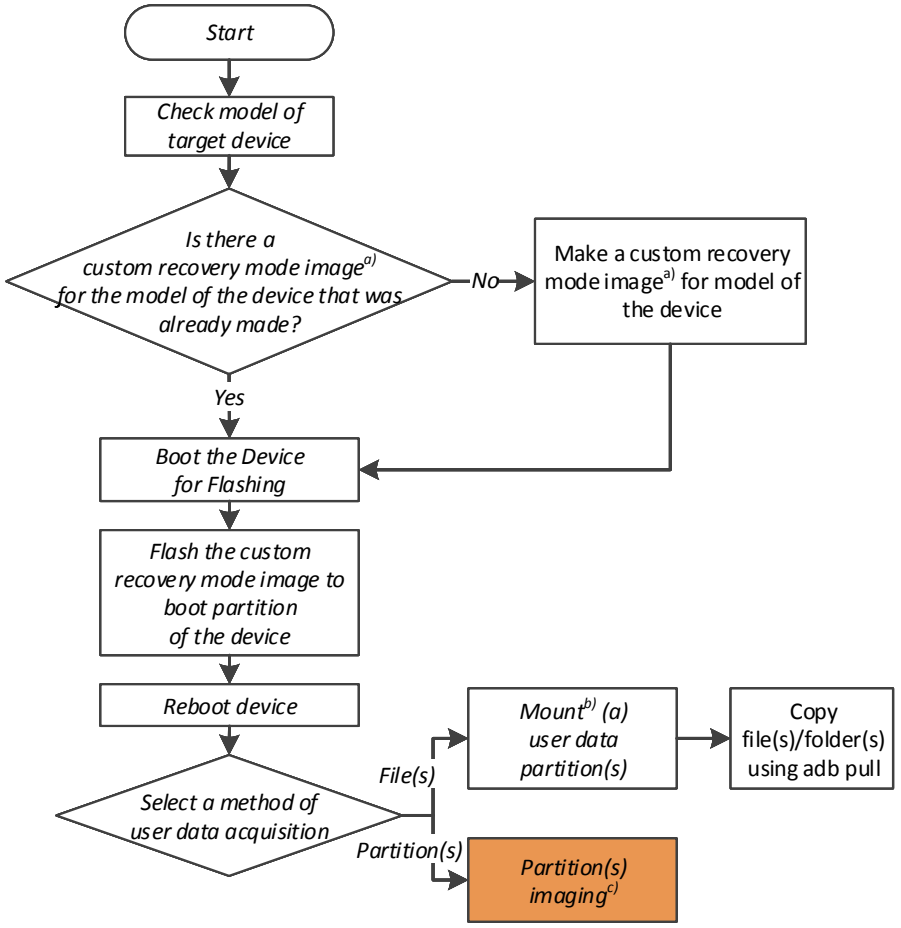
2. **If select a file acquisition**

   - Mount a user data partition (read-only)
   - Copy files in data partition using ADB protocol

     ex) adb pull /data/data/com.android/databases/xxx.db

3. **If select a partition acquisition**

   - Imaging data/SDcard partition
   - Use DD, NC binary



Return to Former State (5/6)

1. **Check the original firmware version of the device**

   ➔ /system/build.prop file

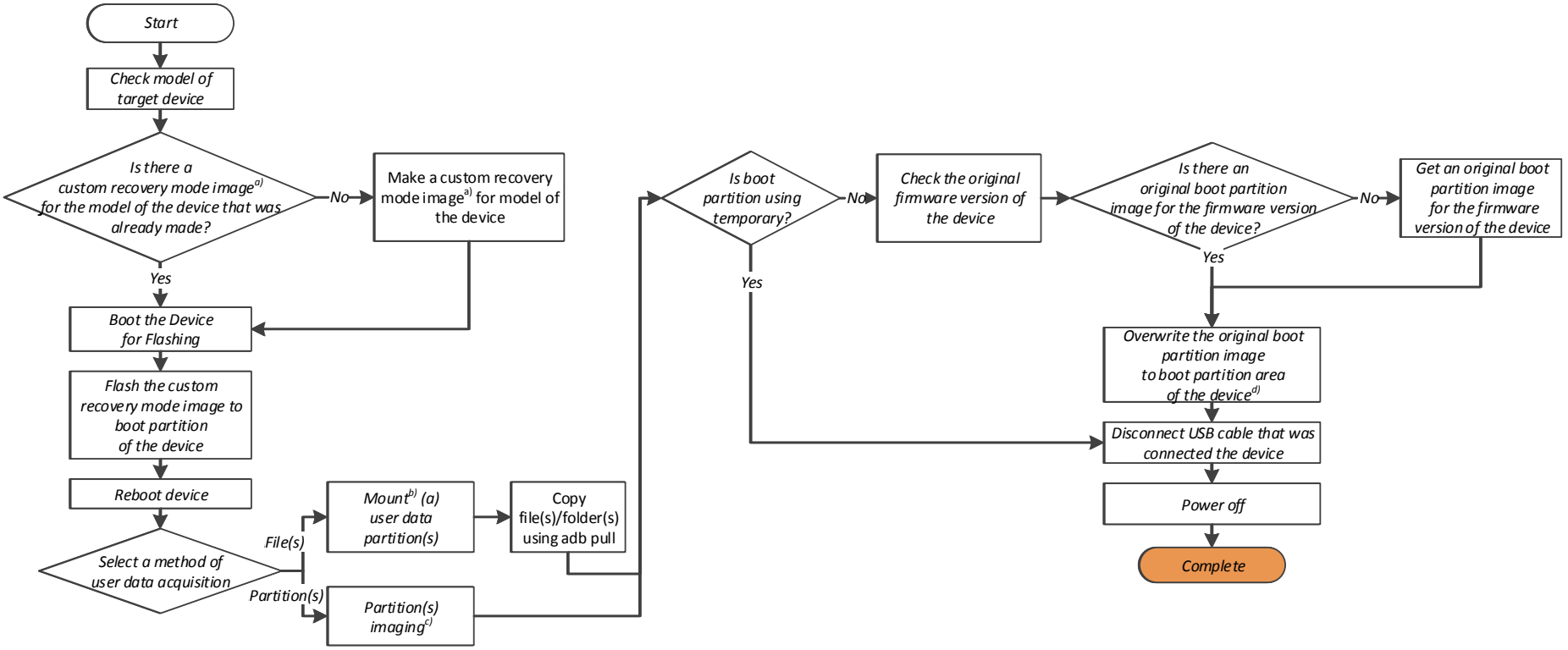
2. **Overwriting the original boot image to boot partition by using DD binary**

   ➔ copy the boot partition to the device by using ADB push

Device	Boot block name
Droid (A855)	/dev/block/mtdblock5
Galaxy S2 (SHW-M250S)	/dev/block/mmcblk0p5
Galaxy Nexus (SHW-M420K)	/dev/block/mmcblk0p7
Galaxy Note (SHV-E160S)	/dev/block/mmcblk0p8
Galaxy S3 (SHV-E210S)	/dev/block/mmcblk0p5
Galaxy Note 2 (SHV-E250S)	/dev/block/mmcblk0p8
Vega LTE (IM-A800S)	/dev/block/mmcblk0p8

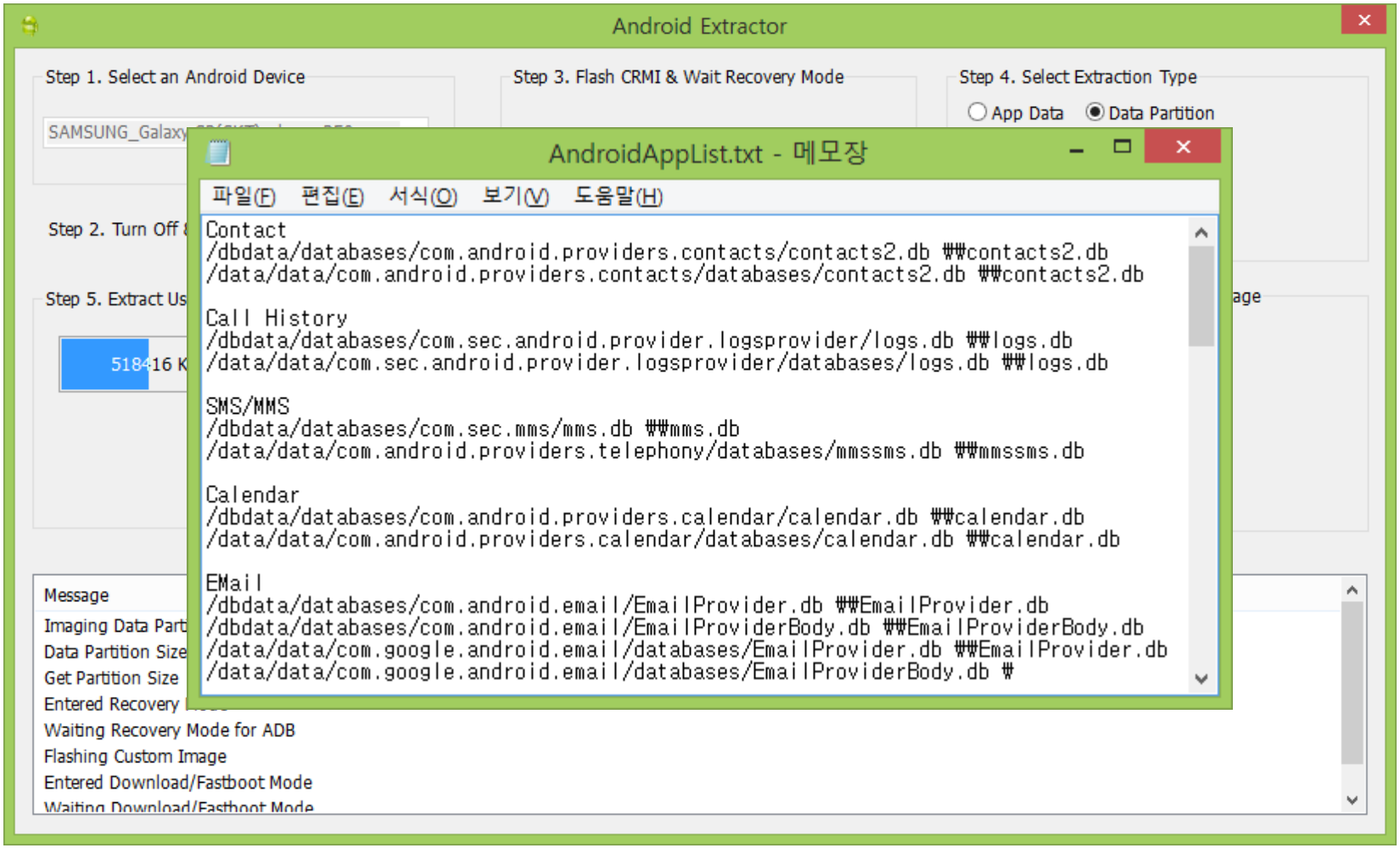
Restoring a Device to Its Original State (6/6)

1. **Disconnect the USB Cable**
2. **Battery remove in order to prevent data modification**
   - Sequence is very important (disconnect the USB cable → remove the battery)
   - Certain devices (ex: Galaxy S2) mount the data partition by only cable power



Android Extractor

Introduce an Android Extractor tool



Experiment using Android Extractor

- **Experiment Method**
- **Experiment Result**

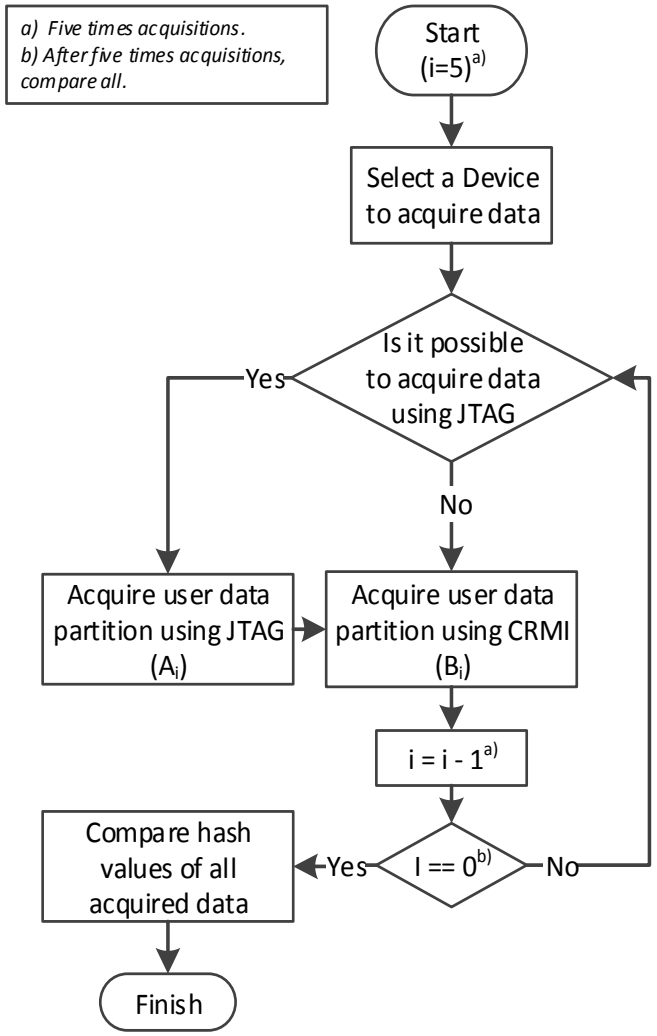
Experiment Method

Test user data integrity during the acquisition process

- **Select seven different Android devices for test**

Device	JTAG to CRMI	CRMI to CRMI
Droid (A855)		√
Vega LTE (IM-A800S)		√
Galaxy S2 (SHW-M250S)		√
Galaxy Nexus (SHW-M420K)		√
Galaxy S3 (SHV-E210S)		√
Galaxy Note (SHV-E160S)	√	
Galaxy Note 2 (SHV-E250S)	√	

- **Repeating the process multiple(5) times**



Experiment Result

- **Confirm the hash value are same in all observations**
  - JTAG to CRMI
  - CRMI to CRMI
- **Suggested data acquisition method preserves the integrity of the data**
  - JTAG also preserve integrity

Demonstration

Conclusion

Conclusion

- **This study explained a method of preserving integrity at the time of user data acquisition**
  - by using the previously studied Recovery Mode

- **Result of the experiment**
  - Method of user data acquisition using CRMI preserve integrity
    - JTAG also preserve integrity of user data
  - Faster than JTAG

Conclusion

- **In order to return to the former state after flashing the CRMI and acquiring data**
  - Need an original boot partition from the firmware version
  - If you do not have an original boot partition
    - Flashing the CRMI to recovery partition instead boot partition

- **There are several limitations, but..**
  - The CRMI method is more efficient
    - compared to other existing methods of forensically sound data acquisition from Android devices.

Q & A

http://forensic.korea.ac.kr