brightsight®

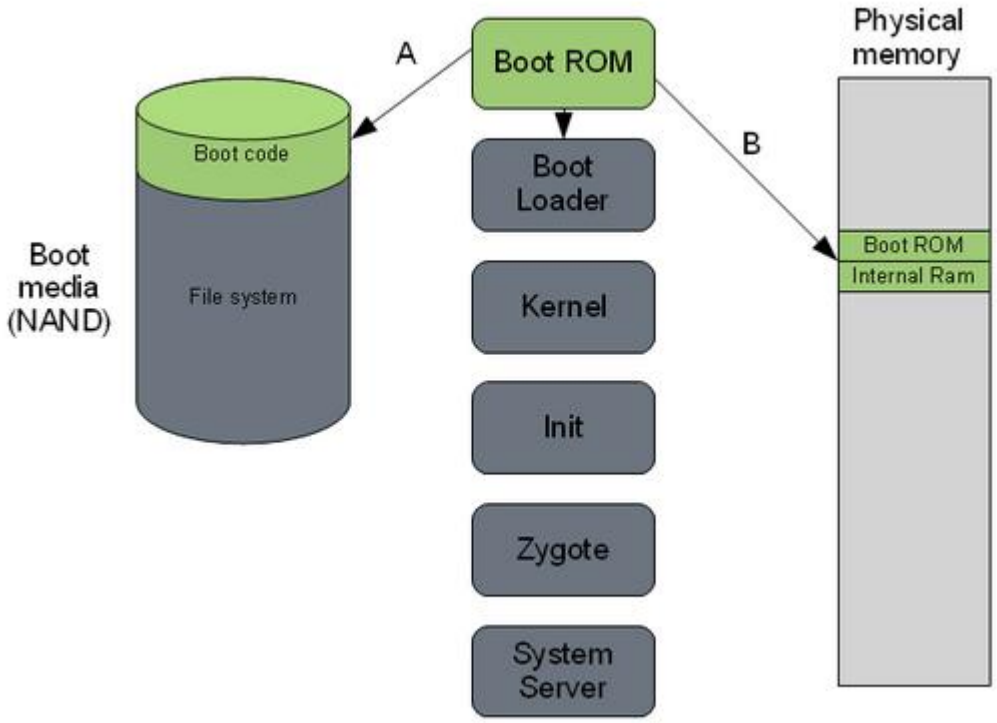
the number one
security lab
in the world

Android Security

Secure Booting (B1)

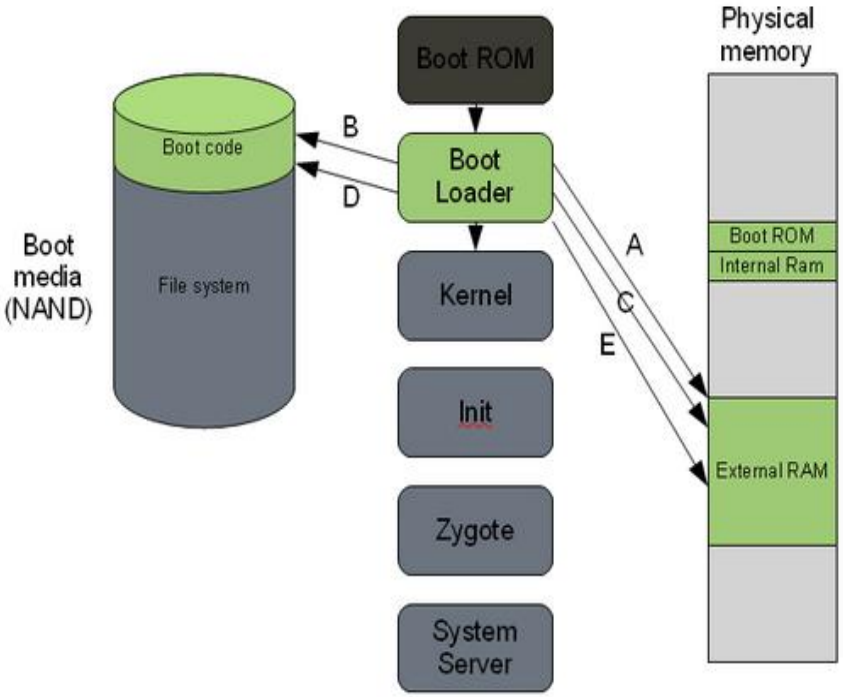
Shiqi Li / Brightsight

Booting process in Android



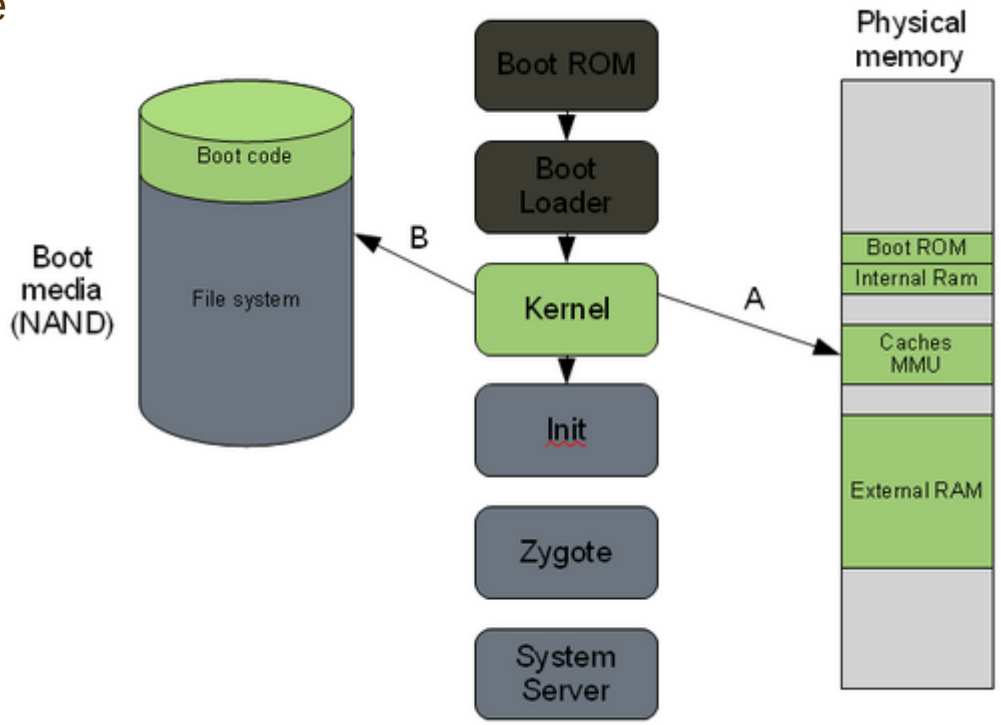
BootLoader

- Code that is executed to boot up Operating System Android
- Low-level system initialization
- Hardware specific
- Security checkpoint
- Support includes:
  - Loading recovery images
  - Loading system flash
  - Performing updates



From Linux → Android

- Kernel of hardware, driver and file system initialization,
- Init: starting processes
- Zygote: creates virtual machine
- System Server: Android



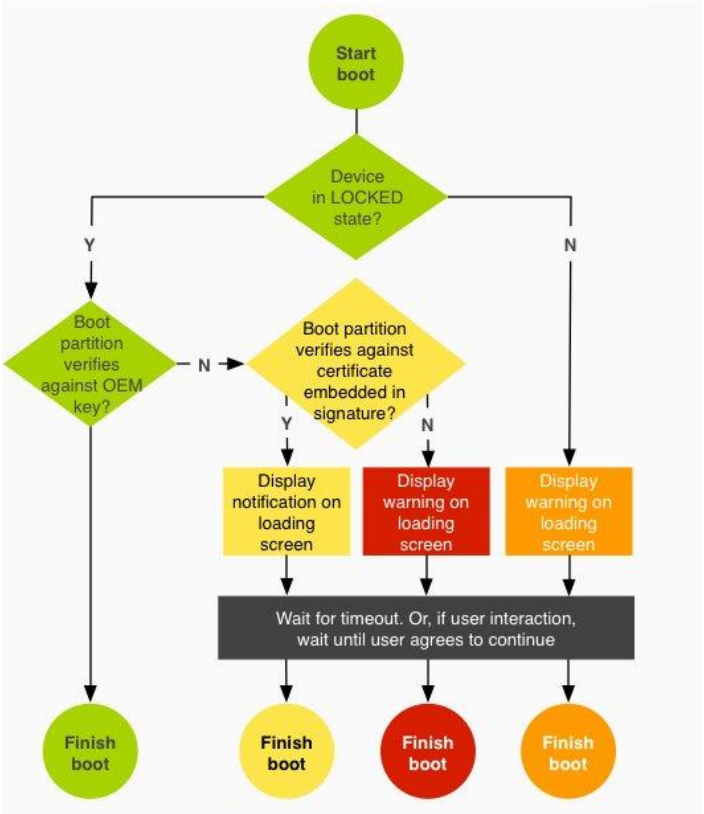
Flash partitions

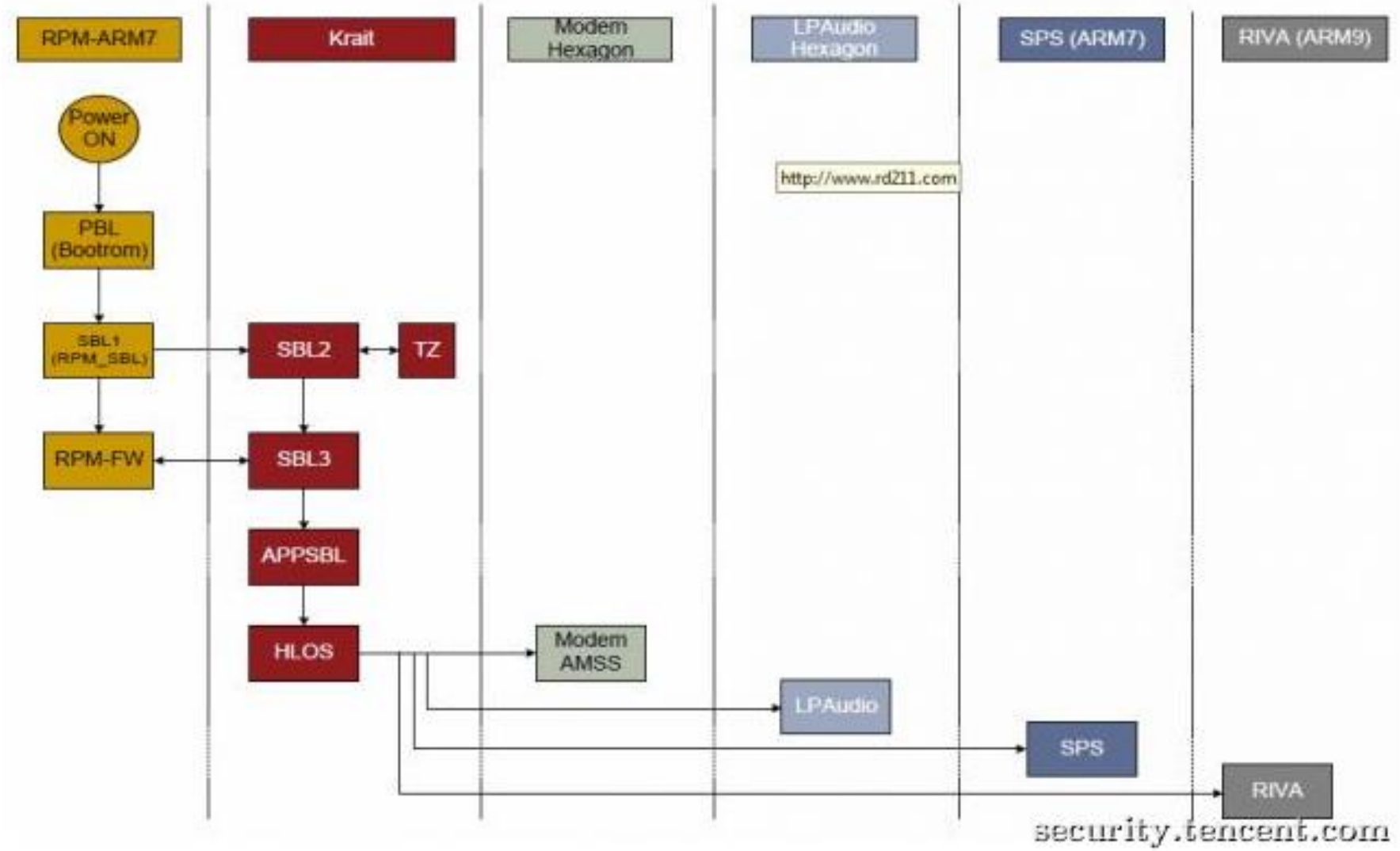
splash1	splash screen image.
misc	misc - has apparently some flags used for controlling device mode
recovery	kernel, initrd with rootfs (for alternate boot) => flashing can be used as a first step to root devices
boot	kernel, initrd with rootfs (for default boot)
system	yaffs2 file system, mounted read-only at /system - has the bulk of the Android system, including system libraries, Dalvik and pre-installed applications.
cache	yaffs2 file system, mounted at /cache - used on G1 for over-the-air updates. This partition can be used to store temporary data.
userdata	yaffs2 file system, mounted at /data - contains user-installed applications and data, including customization data

Verified Boot

Booting via chain of trusted verified partitions

- Boot stage: Red, Yellow, Green, Orange
- Device: locked or unlocked.
- Fixed OEM key
- dm-verity performs the check (integrity on block devices)




RPM-ARM7
Krait
Modem Hexagon
LPAudio Hexagon
SPS (ARM7)
RIVA (ARM9)
Power ON
http://www.rd211.com
PBL (Bootrom)
SBL1 (RPM_SBL)
SBL2
TZ
RPM-FW
SBL3
APPSBL
HLOS
Modem AMSS
LPAudio
SPS
RIVA
security.tencent.com

Rooting

Flash storage system contains:

- /data → android applications
- /system → operating system
  - Normally read-only
  - Normally not accessible for an android user
  - ➔ only root can read / write.

How to tell that an android device is rooted:

- Android devices:
  - # means the device is rooted
  - $ means you are a normal user
- SU or superuser is installed
- /sbin directory is available

Becoming Root:

- Flashing a recovery partition
- Using Vulerabitily

Evaluation check points

- What firmware/partition can be flashed?
  - Multiple processors
- What are the security mechanisms used to protect boot partitions?
- How is the device flashed (OTA, fastboot, USB)?
- How is the firmware of each boot stage verified (which key, key length, hash algorithm)
- Settings/configuration related to “secure booting”

PCI PTS security concerns

DTR B1:

- How does the device perform self-test during boot up? How is it initiated?
- How to handle self test failure and fail in secure manner
- Boot chain of device
- The provided source code shall show
  - self-test every 24 hours/prior to PIN entry
  - self-test / verification functions are present
  - Proper algorithms, register settings are used correctly
  - Handling of self-test fails
  - Memory is re-initialisation every 24 hours
- How are the authenticate cryptographic keys protected
- Identify and show that how each piece of boot image and firmware components are authenticated during booting (e.g. filesystem or executable/library)
  - V4.1 also requires authenticated applications to be included in the selftest process
- Confirm self-test includes registers settings relied on for security

PCI PTS security concerns

DTR B4/B4.1

The evidences shall

- show that the device cryptographically authenticates firmware/application integrity
- show that the device authenticate external components for FW/SW update
- show that device rejects unauthorized firmware/application
- show which component performs authentication of firmware/application
- Controls provide for unique accountability and utilize key sizes appropriate for algorithms
- provide complete table of processing elements (as also given in A4)
- show, If applicable, detail various types update images differentiated from each other
- show in source code that
  - FW/SW are authenticated by secure firmware
  - if HMAC is used no leaking of timing information
  - if CBD MAC used, detail method to mitigate vulnerabilities
- show how Public keys are loaded during manufacturing, and how default values are changed

Questions?