Secure Boot, Chain of Trust and Data Protection

Akshay Bhat

embeddedworld2019
Exhibition&Conference
... it´s a smarter world

Topics

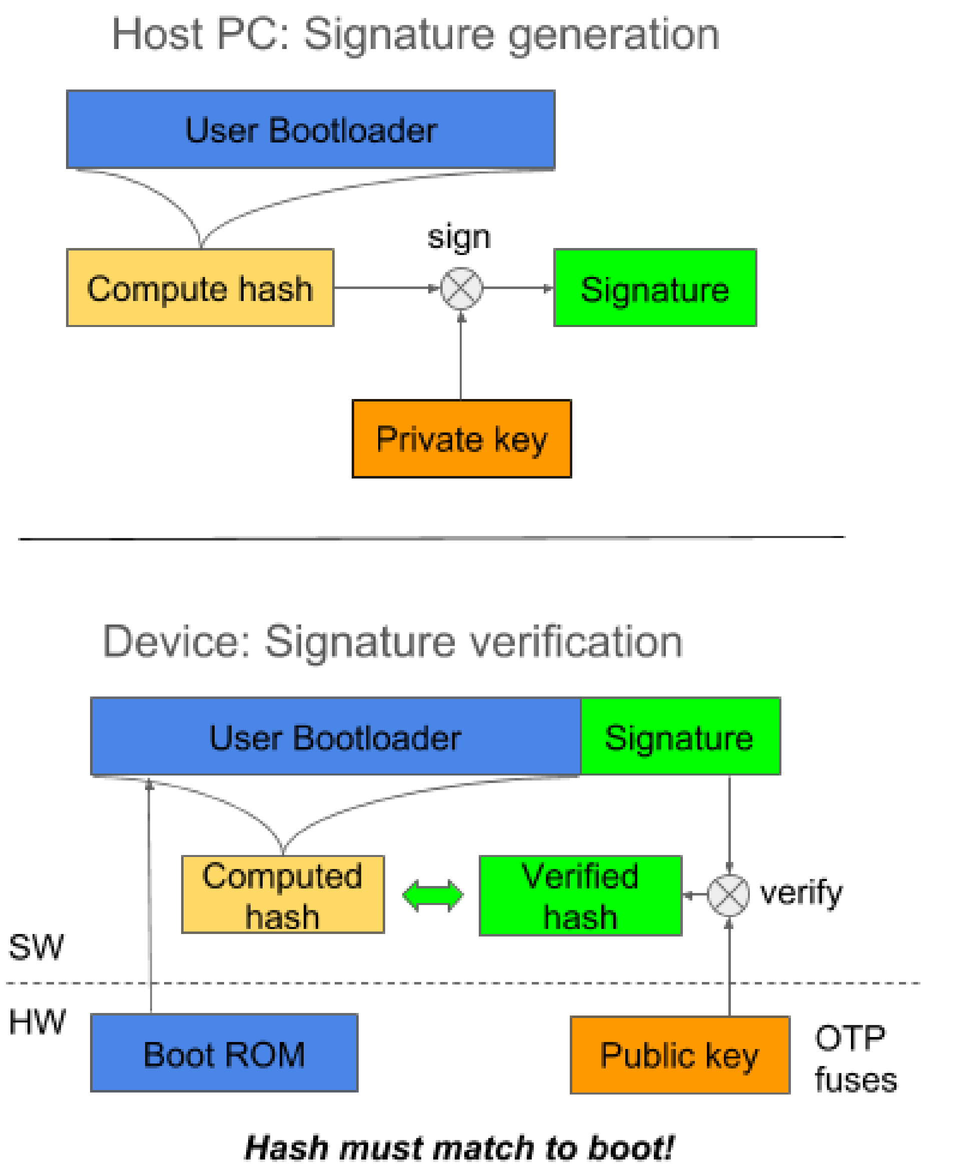
- **Introduction to secure boot**
- **Chain of trust**
- **Protecting data**
  - Secure key storage
- **Best practices and lessons learnt**

Secure boot overview

- **Provides**
  - Authentication (unauthorized images not allowed to run)
  - Integrity (authorized images can not be ‘tampered’ with)
- **Digital signatures for authentication**
  - Private key -> used for signing
  - Public key -> used to verify
- **Image/data encryption**
  - Confidentiality
  - Anti-cloning/counterfeit
    - Unique keys required

Bootloader Authentication

- **Microprocessors**
  - Performed by built-in ROM code
- **Microcontrollers**
  - User implemented code (eg: mbed TLS)
    - Flash locked from modification



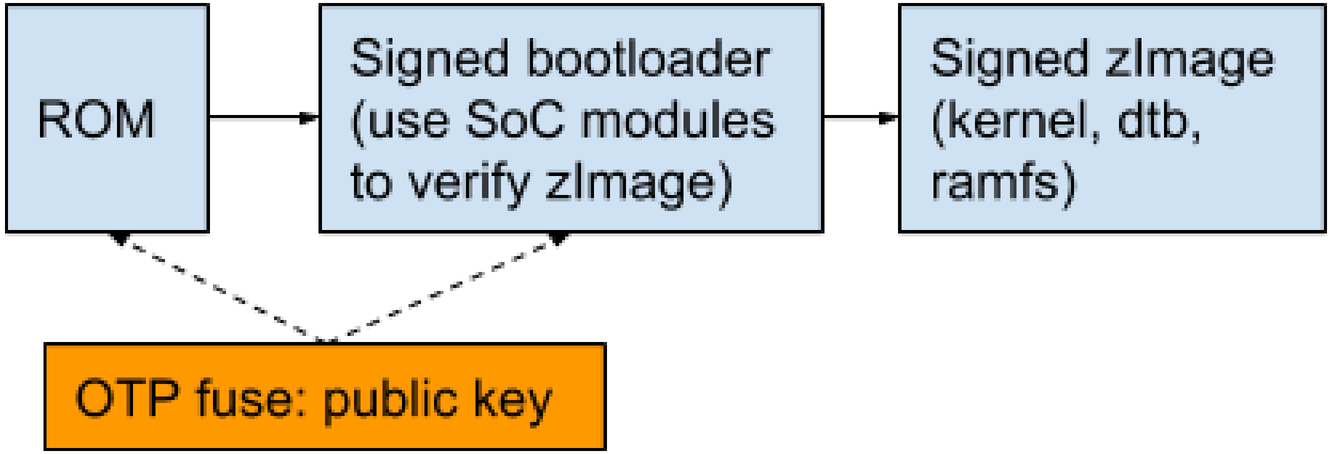
Hash must match to boot!

Components of Linux device

- **Bootloader**
  - First stage (eg: SPL, SBL, ARM-TF)
  - Second stage (eg: u-boot, barebox, little kernel)
- **Kernel**
- **Device tree**
- **Root filesystem**
  - User data partition
- **Optional**
  - Secure OS (eg: op-tee)
  - Firmware (eg: FPGA, FreeRTOS on M3/M4)

Chain of trust

- **SoC specific mechanism extended**



ROM

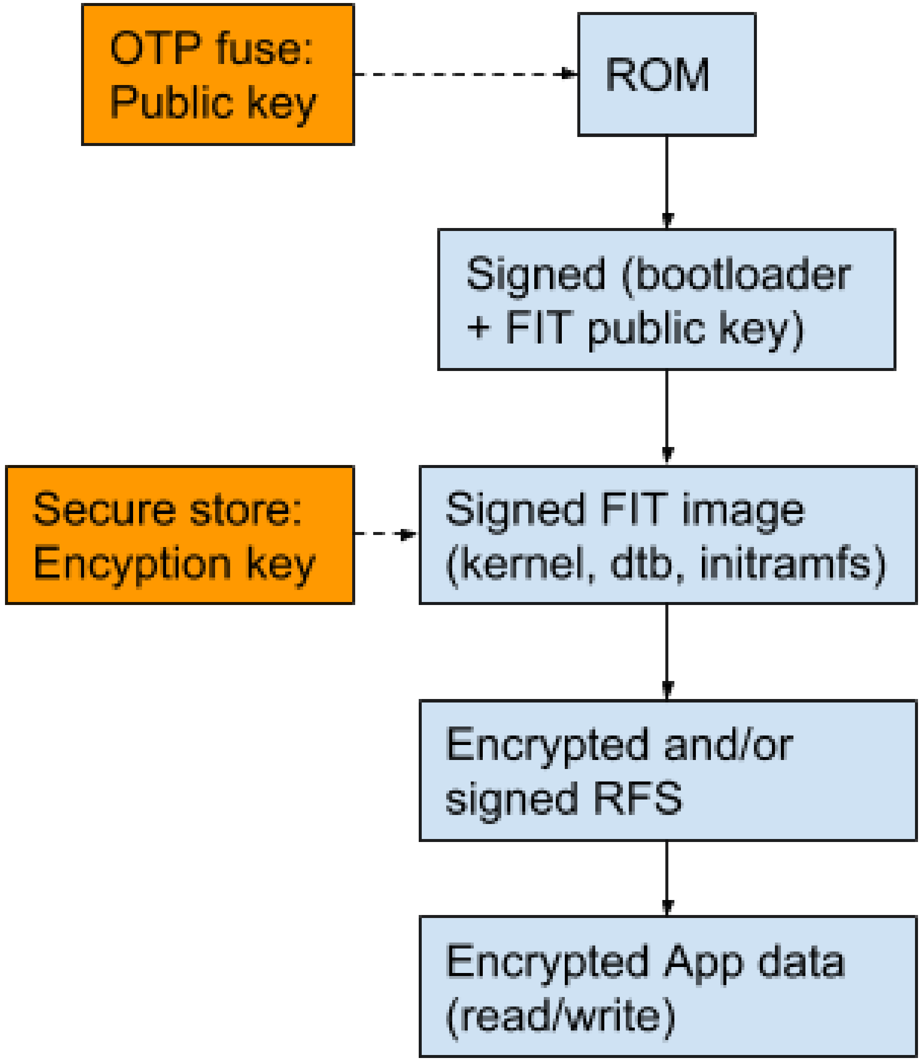
Signed bootloader (use SoC modules to verify zImage)

Signed zImage (kernel, dtb, ramfs)

OTP fuse: public key

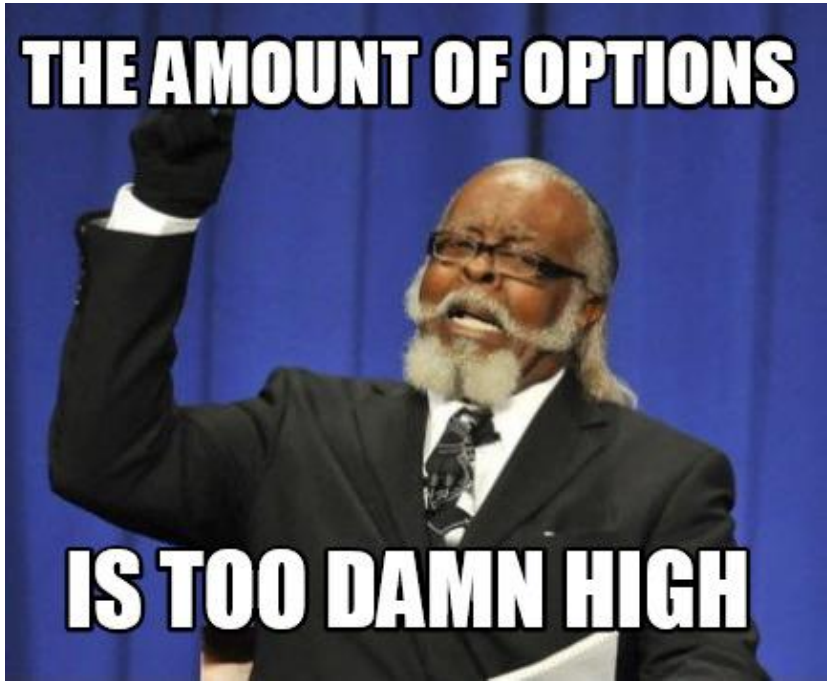
Chain of trust

- **Open source mechanisms**
- **FIT (Flattened Image Tree) option in u-boot**


OTP fuse: Public key
ROM
Signed (bootloader + FIT public key)
Secure store: Encyption key
Signed FIT image (kernel, dtb, initramfs)
Encrypted and/or signed RFS
Encrypted App data (read/write)

Protecting userspace components

- **Block level**
  - dm-crypt (encrypted)
  - dm-verity (signed – read only)
  - dm-integrity (encrypted and authenticated)
- **Filesystem level**
  - fscrypt (ext4, ubifs etc)
  - ecryptfs



Secure key storage

- **No user input on most devices**
- **SoC specific mechanism**
  - Keys stored in secure fuses (OR)
  - Keys encrypted using unique master key (eg: i.MX)
- **Trusted Execution Environment**
  - ARM TrustZone
- **TPM**
  - Seal keys using PCR registers
- **Crypto chip**
  - Beware of I2C bus attacks

Additional Security Measures

- **Hardware security**
  - JTAG
  - Tamper protection
- **Known vulnerabilities**
  - Processor specific (eg: CVE-2017-7936)
  - Bootloader specific (eg: CVE-2018-18439)
- **Secure OTA update process**
  - Signed and/encrypted OTA images
  - Server authentication

Other considerations

- **Trade-offs**
  - Boot time
  - Filesystem performance
- **Securing the private and encryption keys**
  - Consider dedicated signing server
- **Key revocation strategy**

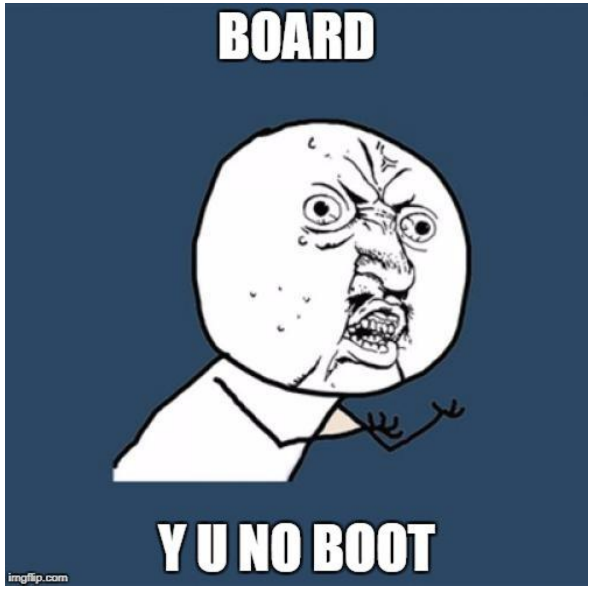
Design documents and Test plan

List of software components, protection mechanism

Component	Scheme	Crypto	Key storage	Key unique?
U-boot	Signed, vendor	RSA	Public key in OTP	No
Kernel	Signed, openssl	RSA	Public key in u-boot	No
RFS	Encrypted	AES	AES key in OTP	Yes

Negative test cases

- Tampered images
- Unsigned images
- Signed with different key



Hardware considerations

- **Microcontrollers**
  - User programmable flash locked regions
- **Microprocessors**
  - ROM support for secure boot
- **Nice to have**
  - Secure key storage
  - Key revocation
  - Hardware accelerated ciphers
  - Customer programmable keys
  - Easy access to signing tools
  - Tamper protection

Take away

- Design in security early
- Select the right hardware components
- Implement security at all software layers
- Continue to monitor vulnerabilities

Questions ?

Thank you

Visit us:

STMicroelectronics Booth

Hall 4A | Stand 138