Overview of Secure Boot state in the ARM-based SoCs

Hardware-Aided Trusted Computing devroom

FOSDEM 2021

Maciej Pijanowski

3MDEB

Agenda

- whoami
- Company profile
- What do we mean by Secure Boot
- Typical implementation
- Typical workflow
- Hardware examples
  - availability overview
- Research results
- Summary
- Contact us
- Q&A

/usr/bin/whoami

Maciej Pijanowski
Embedded Firmware Team Leader

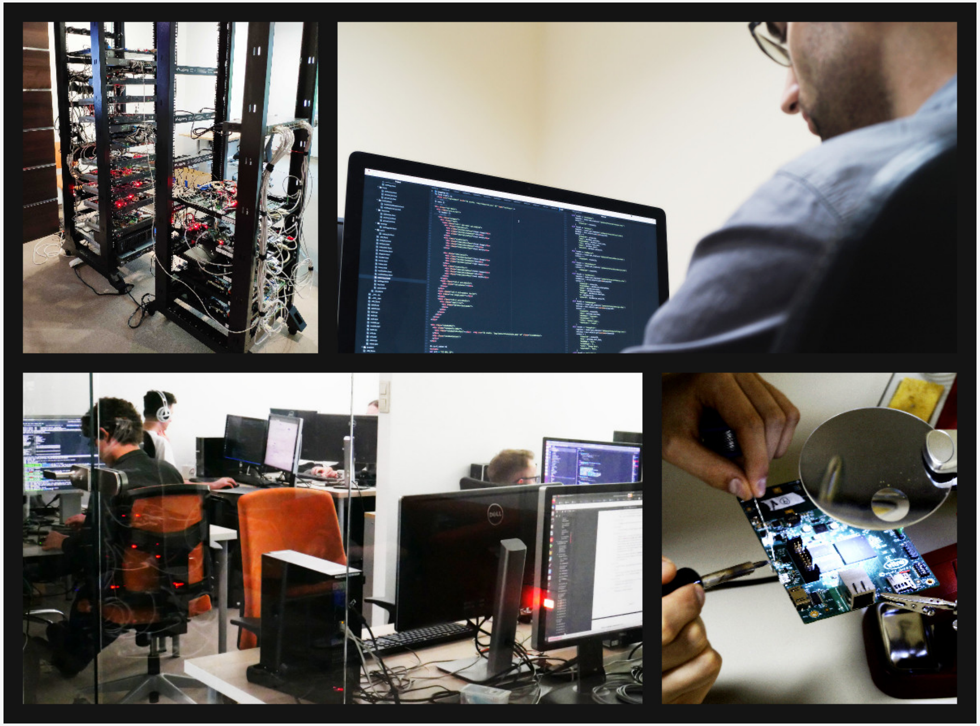
- @macpijan
- maciej.pijanowski@3mdeb.com
- linkedin.com/in/maciej-pijanowski-9868ab120

- 5 years in 3mdeb
- interested in:
  - Embedded Linux
  - build systems (e.g. Yocto)
  - system security

Company profile

3mdeb is a firmware and embedded systems development company founded by Piotr Król and headquartered in Gdańsk, Poland. We combine the collaborative creativity of the open-source community with the reliable strength of an enterprise-grade solutions provider.

Company profile



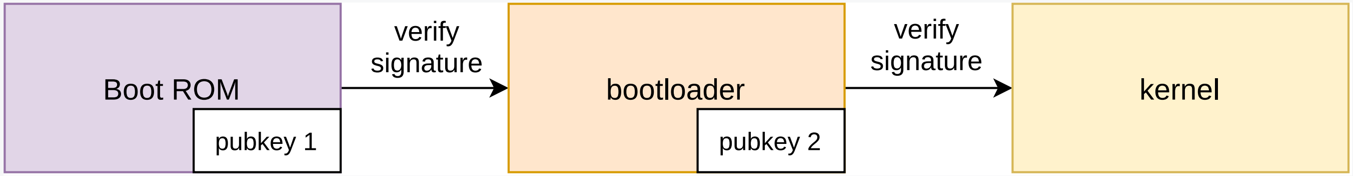
Our team is made up of engineers with vast experience working with UEFI/BIOS, coreboot, Linux, Yocto, and more. We create IoT and firmware solutions supporting security and integrity standards; roots of trust, boot integrity, TPM, DRTM, and much more.

What do we mean by Secure Boot

- We focus on the ARM context in this presentation
- Boot ROM feature
- Verified Boot
- To verify the firmware before executing it
    - verify the signature
    - private key was used to sign the binary
    - public key must be known by the device
- Boot ROM is assumed to be trusted
    - closed source
- The meaning of Secure Boot for different architecture can be different

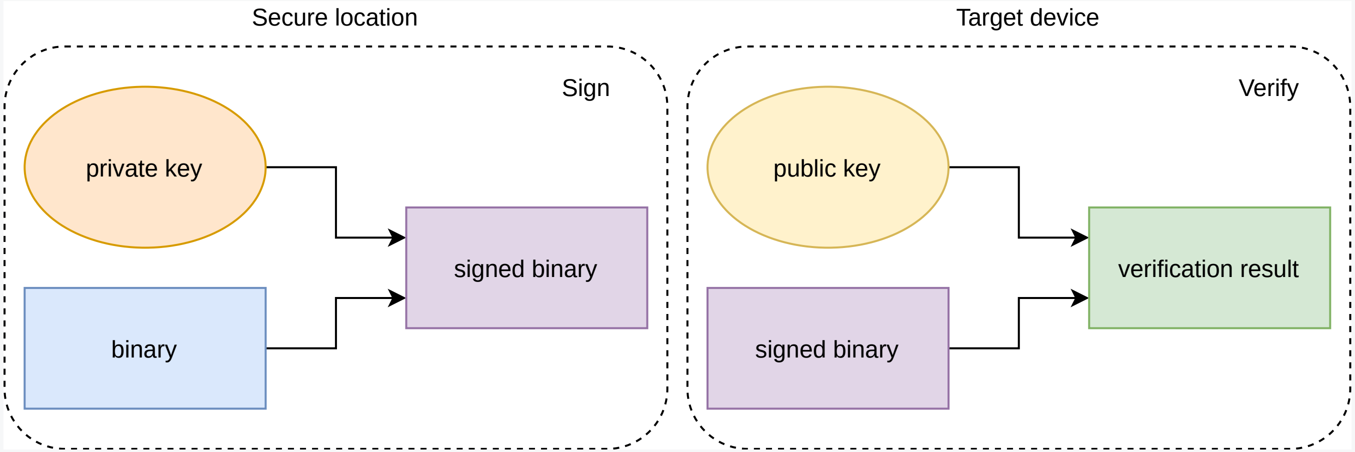
Typical implementation

- Public key written into the SoC
  - electrical Fuse (eFuse)
  - OTP (One-Time-Programmable) registers
  - Root of Trust
- Next components can use different keys
  - must be locked down (e.g. disabled U-Boot shell)
  - to preserve the chain of trust
- We are focusing on the first step
  - the verification of the first binary executed by the BootROM



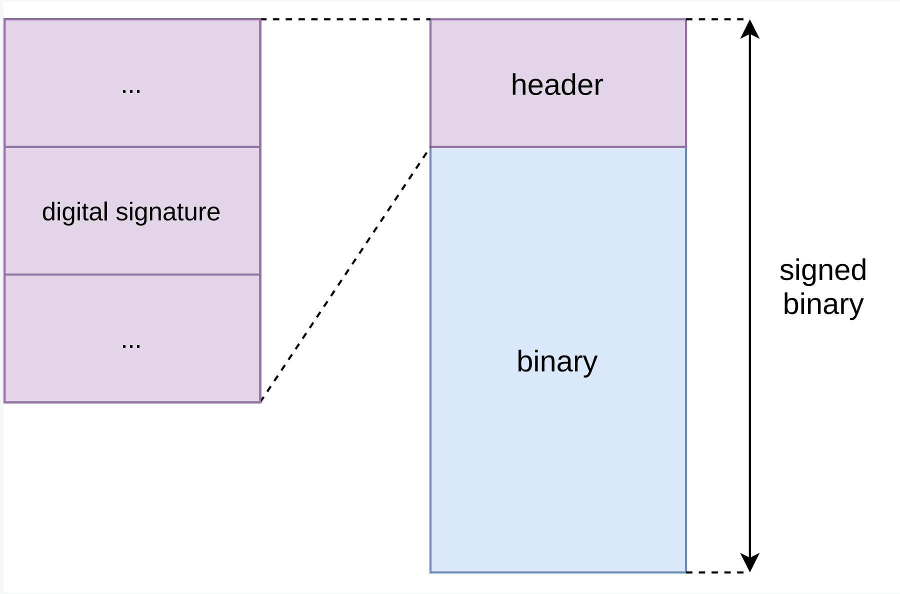
Typical workflow

- Generate keypair
- Sign the firmware binary
- Fuse the public key into the SoC
- Enable Secure Boot feature
- Confirm whether the firmware verification works correcly
- Close (lock) the platform
    - at this point only the signed firmware can be executed



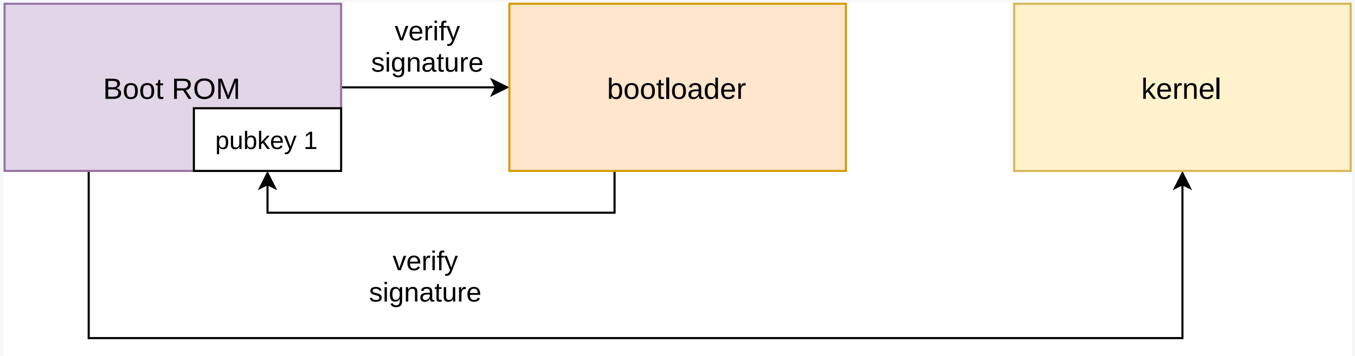
Typical workflow

- Signed binary layout
- Typically original data extended with some header
  - specific to the given implementation
  - digital signature is here



NXP - i.MX 6/7/8

- HABv4 (High Assurance Boot)
  - Boot ROM feature
- HABv4 features
  - store up to 5 public keys
  - RSA 1024-4096 keys
  - EC p256, p384, p521 keys
- HAB API can be used to verify more than one binary



NXP - i.MX 6/7/8

- Application notes freely available
  - AN4581 i.MX Secure Boot on HABv4 Supported Devices: https://www.nxp.com/docs/en/application-note/AN4581.pdf
  - AN12263 HABv4 RVT Guidelines and Recommendations: https://www.nxp.com/docs/en/application-note/AN12263.pdf
- Lot's of guides online
- SoC Reference Manual available after free registration
- Signing tool
  - Code Signing Tool can be downloaded after free registration
  - can be used with HSM
  - AN12812 Using Code-Signing Tool with Hardware Security Module: https://www.nxp.com/docs/en/application-note/AN12812.pdf
- Vulnerabilities in Boot ROM may happen
  - https://blog.quarkslab.com/vulnerabilities-in-high-assurance-boot-of-nxp-imx-microprocessors.html

NXP - Layerscape

- QorIQ Trust Architecture
  - Secure Boot is one of the features
  - similar to HAB
- Application notes freely available
  - AN5227 Secure Boot and Secure Debug Configuration for LS1: https://bit.ly/39Ez3Mm
- Excellent documentation in the LSDK User Guide: https://www.nxp.com/docs/en/user-guide/LSDKUG_Rev20.04.pdf
  - chapter: 6.1 Secure boot
- SoC and Security Reference Manual available after free registration
- Signing tool
  - Code Signing Tool as part of the LSDK

Marvell Armada

- Details not available publicly
- Documentation under NDA
  - SoC Reference Manual under NDA
  - AN-383: ARMADA ® 38x/39x Families Secure Boot Mode Support
- Marvell github repositories provide useful information
  - https://github.com/MarvellEmbeddedProcessors/u-boot-marvell
  - https://github.com/MarvellEmbeddedProcessors/atf-marvell
  - https://github.com/MarvellEmbeddedProcessors/edk2
- Almost step-by-step guide in U-Boot repository
  - https://github.com/MarvellEmbeddedProcessors/u-boot-marvell/blob/u-boot-2018.03-armada-18.12/doc/mvebu/trusted_boot.txt
- The process is similar for 32-bit Armada 38x and 64-bit Armada 7k/8k
- Fuse command available in U-Boot and edk2
- Signing tool
  - tools/doimage in atf-marvell repository

STM32 - STM32MP1 SoC

- Only EC keys are supported
- Quality documentation and guides available on-line
  - https://wiki.st.com/stm32mpu/wiki/Security_overview#Secure_boot
  - https://wiki.st.com/stm32mpu/wiki/STM32MP15_secure_boot
- Key generation
  - https://wiki.st.com/stm32mpu/wiki/KeyGen_tool
- Signing tool
  - https://wiki.st.com/stm32mpu/wiki/Signing_tool
  - tools are part of the STM32CubeProgrammer toolkit
  - free registration required

Xilinx - Zynq

- 2048-bit RSA key
- Documentation freely available
- Application note - XAPP1175 Secure Boot of Zynq-7000 SoC
  - https://bit.ly/3stH1Ao
  - describes the Secure Boot process preparation in great details
- Zynq-7000 SoC Technical Reference Manual
  - https://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf
  - chapter 32: Device Secure Boot

NVIDIA - Tegra

- 2048-bit RSA key
- Documentation and guides available on-line
  - https://bit.ly/35JjSw6
- Nvidia Xavier SoC Technical Reference Manual
  - https://developer.nvidia.com/embedded/downloads#?tx=%24product,jetson_agx_xavier
  - not much Secure Boot related topics
  - the information from on-line documentation os enough
- Fusing tool
  - part of the JetPack SDK
  - odmfuse.sh script
- Signing tool
  - the flash.sh flashing script has option for binary signing

Microchip - SAMA5

- RSA keys
- Publicly available documentation
  - AN2748 SAMA5D2 Linux® Secure Boot: http://ww1.microchip.com/downloads/en/AppNotes/DS00002748A.pdf
  - some details missing
- Documentation behind NDA
  - AN2435 SAMA5D2 Series Secure Boot Strategy application note
- Fusing tool
  - code signing tools (Secure SAM-BA tools) require NDA
  - usage described in the AN2748
- Signing tool
  - code signing tools (Secure SAM-BA tools) require NDA

Texas Instruments - Sitara SoCs

- Details not publicly available
- Publicly available AM335x Technical Reference Manual
    - https://www.ti.com/lit/ug/spruh73q/spruh73q.pdf
    - it only mentions that Secure Boot is available
    - no details
- High-level Secure Boot overview is available
    - https://www.ti.com/lit/wp/spry305a/spry305a.pdf?ts=1610562359010
    - marketing material
- Contact with TI and/or signing NDA required to get more documentation

Qualcomm

- Limited documentation publicly available
    - Secure Boot and Image Authentication Technical Overview: https://www.qualcomm.com/media/documents/files/secure-boot-and-image-authentication-technical-overview-v2-0.pdf
    - high-level overview
    - RSA keys
- Interesting 3rd party writeups
    - https://www.timesys.com/security/secure-boot-snapdragon-410/
    - https://lineageos.org/engineering/Qualcomm-Firmware/
- Signing tool
    - proprietary tool: sectool
    - requires NDA

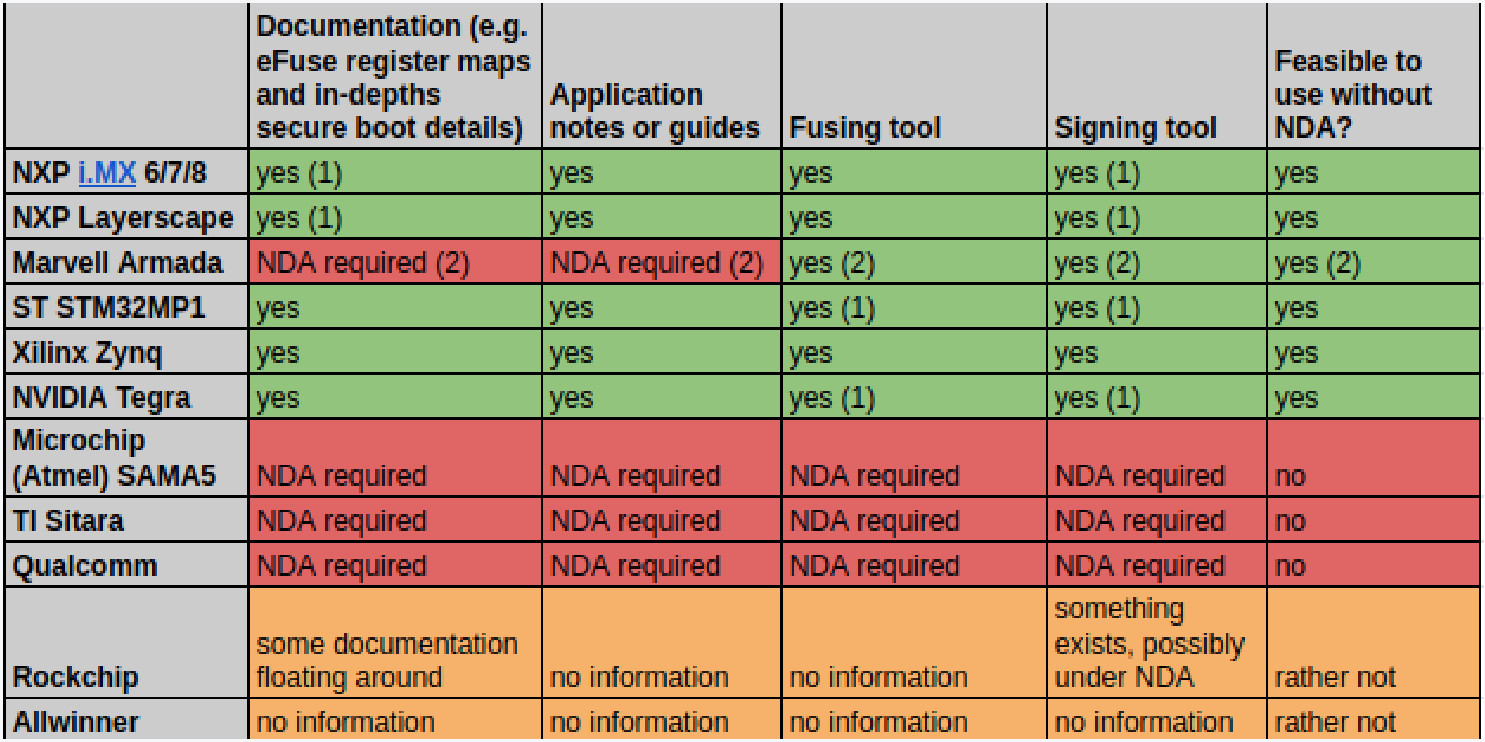
Rockchip

- RK3399 TRM mentions the Secure Boot and eFuse features
    - two 1024bits(32x32) high-density electrical Fuse
    - the document can be found online
    - not clear which eFuse registers store public key
    - not clear how to burn the eFuse
- Some more documents can be found online
    - like the `Rockchip Secure Boot Specification`
    - some Windows tool `SecureBootTool` used to generate keys and sign firmware images
- Most likely some kind of Secure Boot exists, but it is not clear how to use it
    - no success stories found on-line
    - communication with vendor would be necessary

Allwinner

- There are signs that some form of Secure Boot feature is there
- Possibly no one knows how to use that
- The most advanced used story got stucked
    - https://forum.armbian.com/topic/3033-h3-soc-boot-rom-security-e-fuse/page/2/?tab=comments#comment-107012

Overview results



	Documentation (e.g. eFuse register maps and in-depths secure boot details)	Application notes or guides	Fusing tool	Signing tool	Feasible to use without NDA?
NXP i.MX 6/7/8	yes (1)	yes	yes	yes (1)	yes
NXP Layerscape	yes (1)	yes	yes	yes (1)	yes
Marvell Armada	NDA required (2)	NDA required (2)	yes (2)	yes (2)	yes (2)
ST STM32MP1	yes	yes	yes (1)	yes (1)	yes
Xilinx Zynq	yes	yes	yes	yes	yes
NVIDIA Tegra	yes	yes	yes (1)	yes (1)	yes
Microchip (Atmel) SAMA5	NDA required	NDA required	NDA required	NDA required	no
TI Sitara	NDA required	NDA required	NDA required	NDA required	no
Qualcomm	NDA required	NDA required	NDA required	NDA required	no
Rockchip	some documentation floating around	no information	no information	something exists, possibly under NDA	rather not
Allwinner	no information	no information	no information	no information	rather not

]

Overview results

- Comments to the graphic table from previous slide
    - (1) - free registration required
    - (2) - There is a documentation in the Marvell's U-Boot's repository, which should be enough to get things working (but not to fully understand it)

Summary

- The general implementation of Secure Boot is similar across vendors
    - image authentication before execution
    - public key in eFuse
    - burn specific eFuse to "lock" the device (switch to "secure" mode)
- Most commonly RSA-2048 is used as the signing key
    - some support only EC keys (ST)
    - some support both (NXP supports up to RSA-4096 and a few EC keys)
- In all cases the SHA-256 is used as a hash function for digital signature
- Usually also firmware decryption feature is present
    - but it was not in scope of this presentation

Contact us

- We are open to cooperate and discuss
- ✉ contact@3mdeb.com
- ⓕ facebook.com/3mdeb
- @3mdeb_com
- linkedin.com/company/3mdeb
- https://3mdeb.com
- Book a call
- Sign up to the newsletter

Q&A