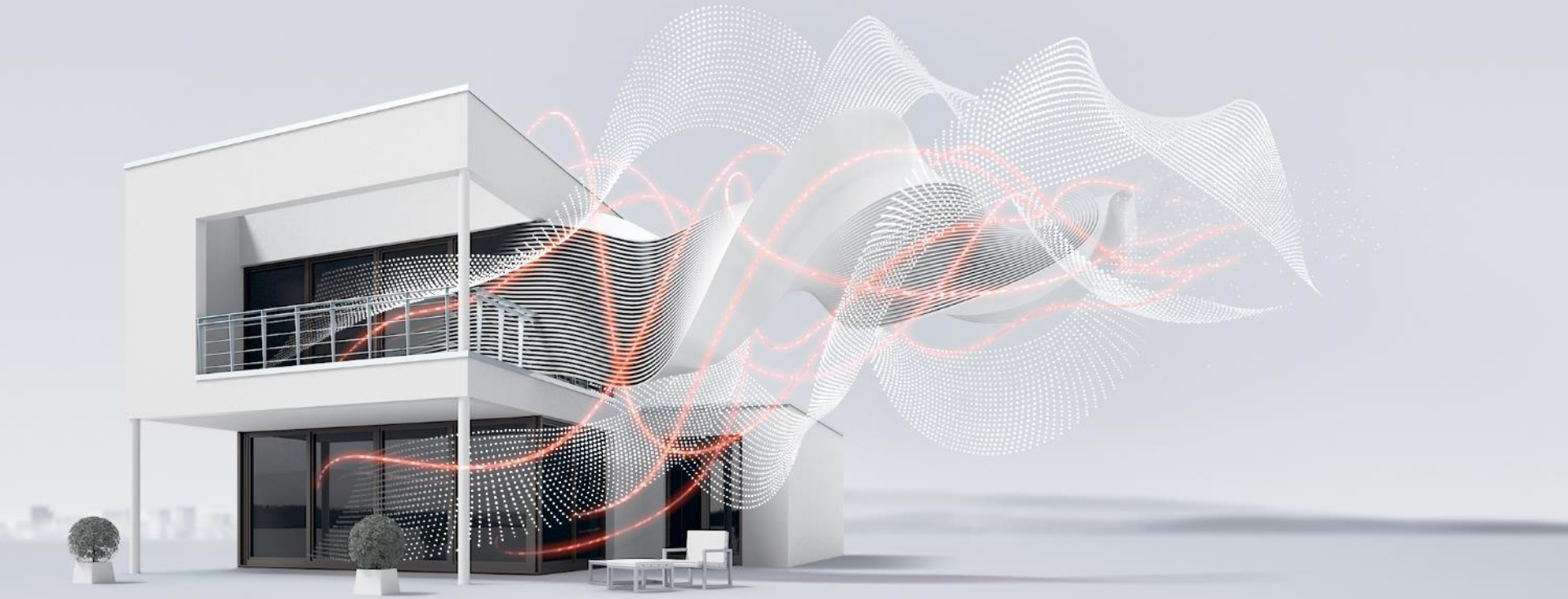


AFIN 2021 - THE THIRTEENTH INTERNATIONAL CONFERENCE ON ADVANCES IN FUTURE INTERNET, ATHENS, GREECE, 2021.11.16

Securing and Hardening Embedded Linux Devices. Theory and Practice

Marcin Bajer, Principal IT Specialist

ABB Corporate Technology Center Kraków

Marcin Bajer

ABB Corporate Technology Center (2015-now)

Firmware development for Smart Buildings products

Jagiellonian University (2012-2013)

Programming Mobile Devices (postgraduate studies)

ABB Corporate Research Center (2006-2015)

Industrial devices development. Motor control & diagnostics

Karel de Grote Hogeschool Antwerpen (2008)

Control Systems Integration using OPC Standard

AGH University of Science and Technology (2003-2008)

Automatics and Robotics (M.Sc.)

Marcin Bajer
Principal IT Specialist
marcin.bajer@pl.abb.com

ABB Corporate Technology Center
Starowiślna 13a
31-038 Kraków, Poland

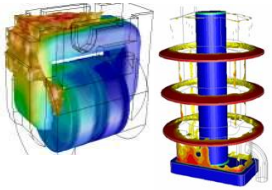
Kraków, Łódź
Corporate Technology Center

- Research
- R&D for BU's
- AI Lab
- Solution accelerators
- Laboratories
- Partnerships: global and local
- Fast prototyping
- Simulations Support Team
- Global Standards Office
- Multidisciplinary teams
- Disruptive entrepreneurial spirit
- Universities
- Start-ups
- Advanced Technology Ventures

ABB Corporate Technology Center

Main competence areas

MV robust insulation systems

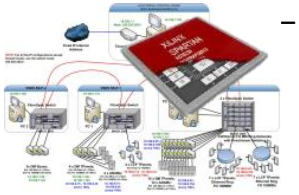


- Functional materials, & Dielectrics, Multiphysical simulations, Manufacturability

Networks & Protection

- Network Analysis
- System & Device Protection

Electronics, Connectivity, software engineering

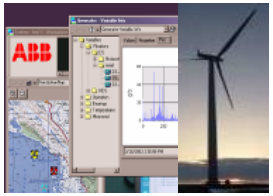


- Embedded Software, Control, FPGA, HF electronics, Analog & Digital, Mobile

Transients, HF Magnetics, EMC & Acoustics

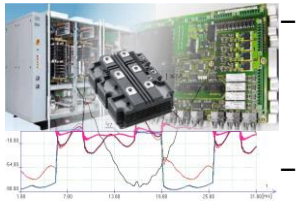
- Passive Design
- Wireless Power Transfer
- Transient mitigation methods

Condition monitoring & Applied analytics



- Algorithms, Systems, Domain knowledge, IoT
- Machine learning
- Physics of failure

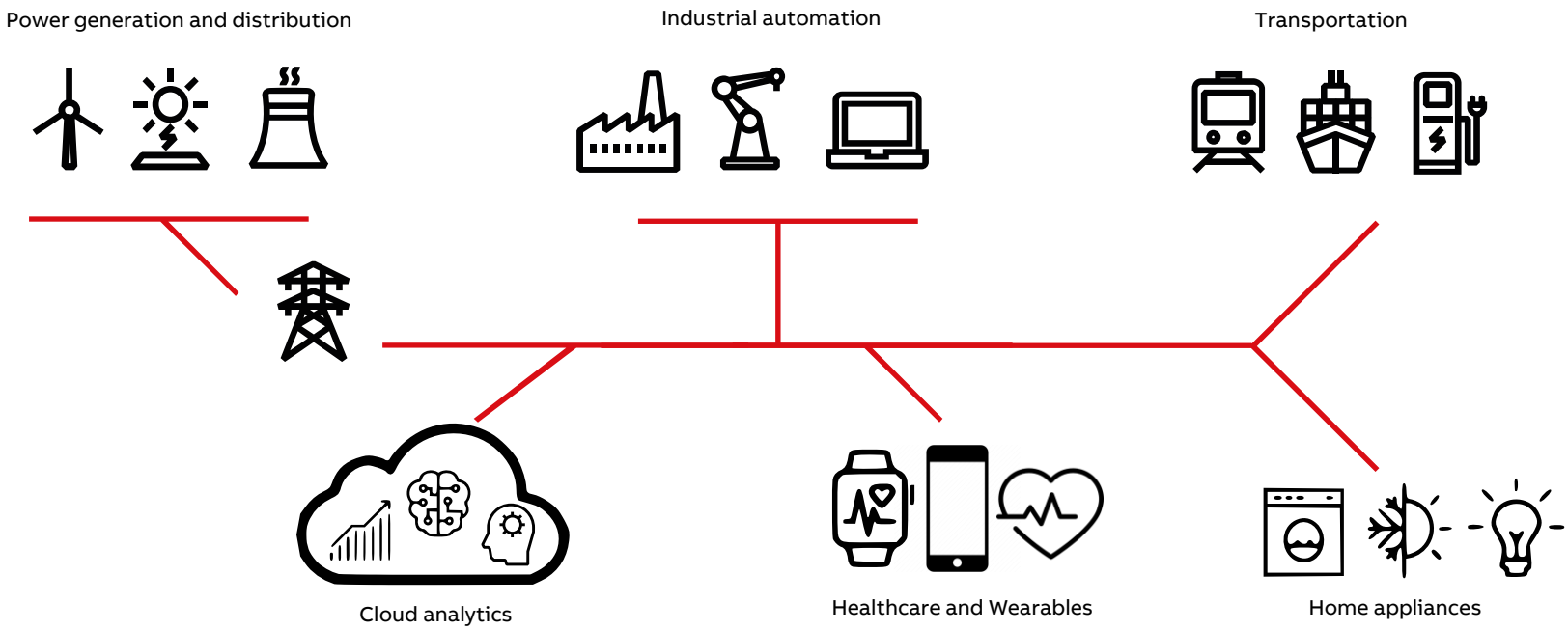
Power electronics/ Solid state switching



- Power Electronics Integration Applications & Reliability
- Active Grid & Switching Systems

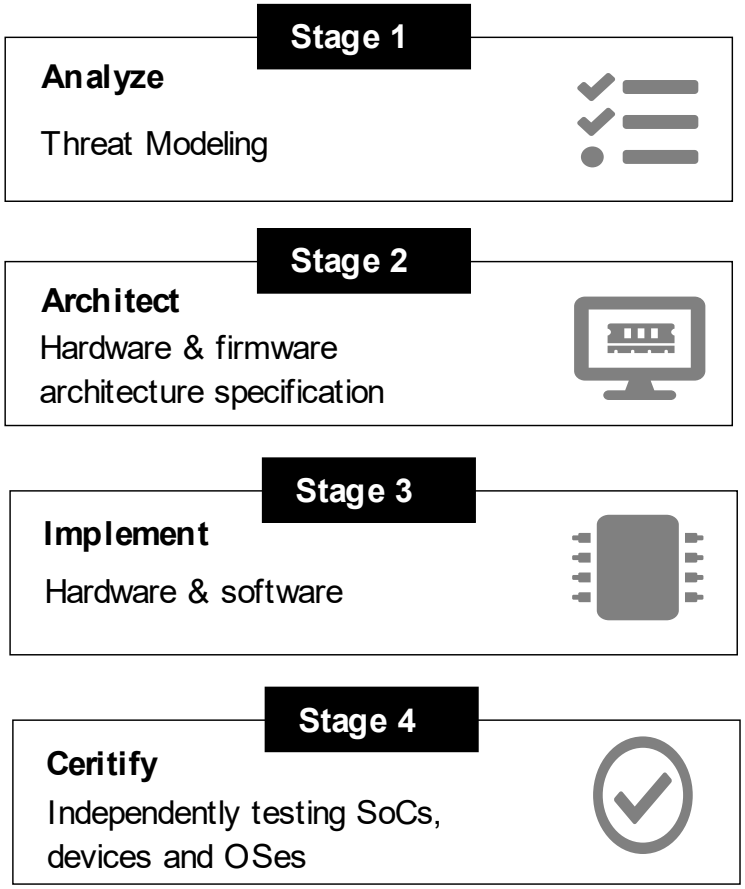
Interconnected world

- Increasing connectivity and complexity,
- Need for comprehensive cybersecurity solutions



Securing embeeded devices

- Not straightforward.
- Continuous process.
- From very beginning of hardware and software development until the very end of it.
- Involves everything from architecture and design throughout the implementation phase till the maintenance.
- Demands from developers a mindset of thinking about the security implications of almost every design decision they made.



Platform Security Architecture (PSA) developed by ARM

Threat modeling

- Threat modeling is one of the very first exercises to perform with the whole product development team.
- The objective of the process is to identify what kind of threats the device might encounter and discuss the assets of the device which need to be protected.
- We have to protect both hardware and software because sometimes the threat agent has a physical access to the device and can be a threat to the assets of the device

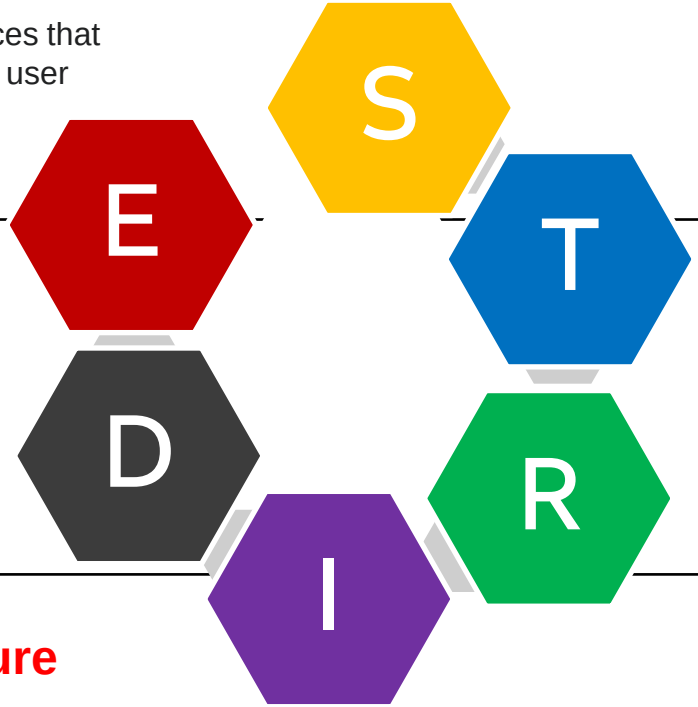
STRIDE Threat Model

Elevation of privilege

Gain elevated access to resources that are normally protected from this user or application

Spoofing identity

Illegally accessing and than using another user's authentication information



Denial of service

Legitimate users are unable to access information systems, devices, or other resources due to the actions of threat actor

Tampering with data

Malicious modification of data. Unauthorized changes

Information disclosure

Exposure of sensitive information to an unauthorized actor

Repudiation

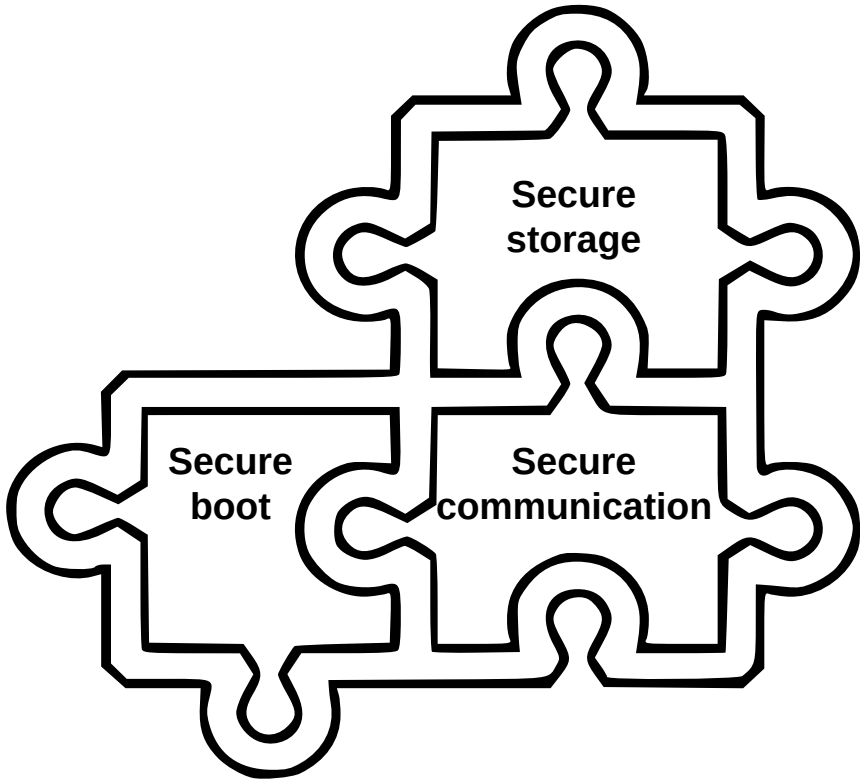
Lack of possibility to prove that user performed an action

Secure enough

- There is no such thing as a fully secure system.
- The development of secure embedded devices is a tradeoff between the expected level of security, costs and functionalities.
- Implementing inadequately high-security measures might affect usability.
- Prioritize the tasks which have the highest impact on device security, protect the most important assets, and mitigate the threats which have the highest possibility to occur.

System architecture

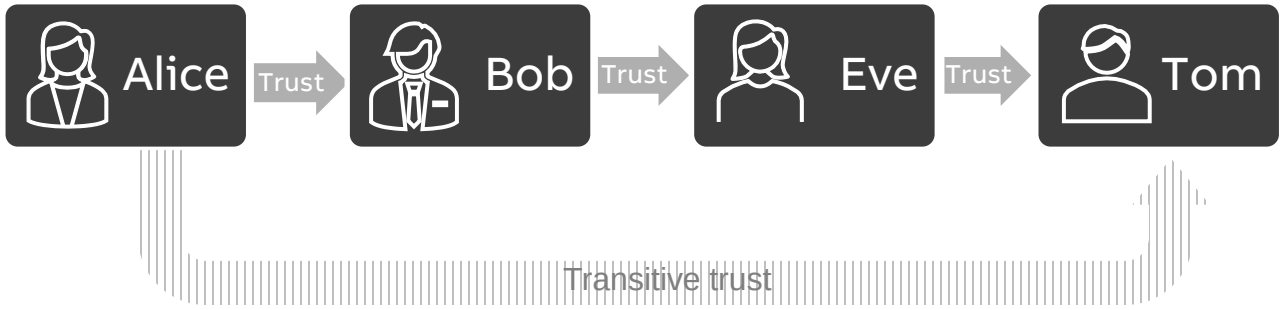
- Required hardware need to be identified (i.e. hardware isolation, crypto engines).
- Technologies, frameworks and architectural design patterns to be used
- Derived requirements such as trusted boot, firmware updates, secure production and device provisioning



Secure boot

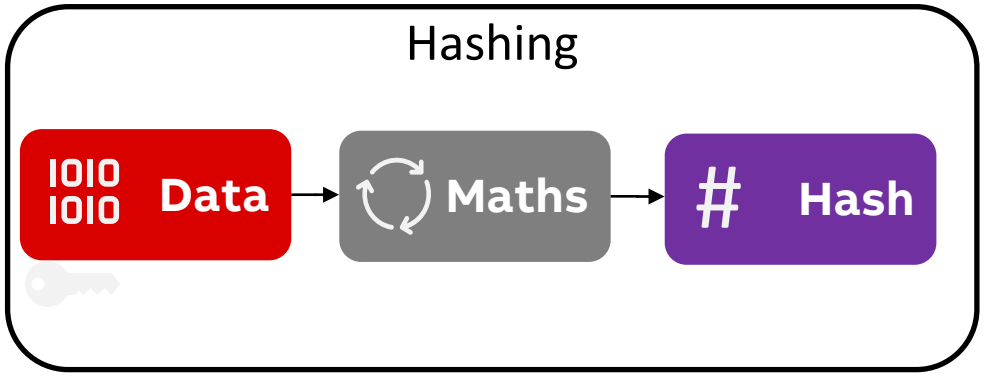
- Multiple terms related to securing device boot process: Trusted Boot, Secure Boot, Measured Boot, Verified Boot.
- Details of implementation specific to CPU architecture and operating system.
- Common principle - maintaining a chain of trust across different layers of software using so-called transitive trust principle.

Transitive trust principle

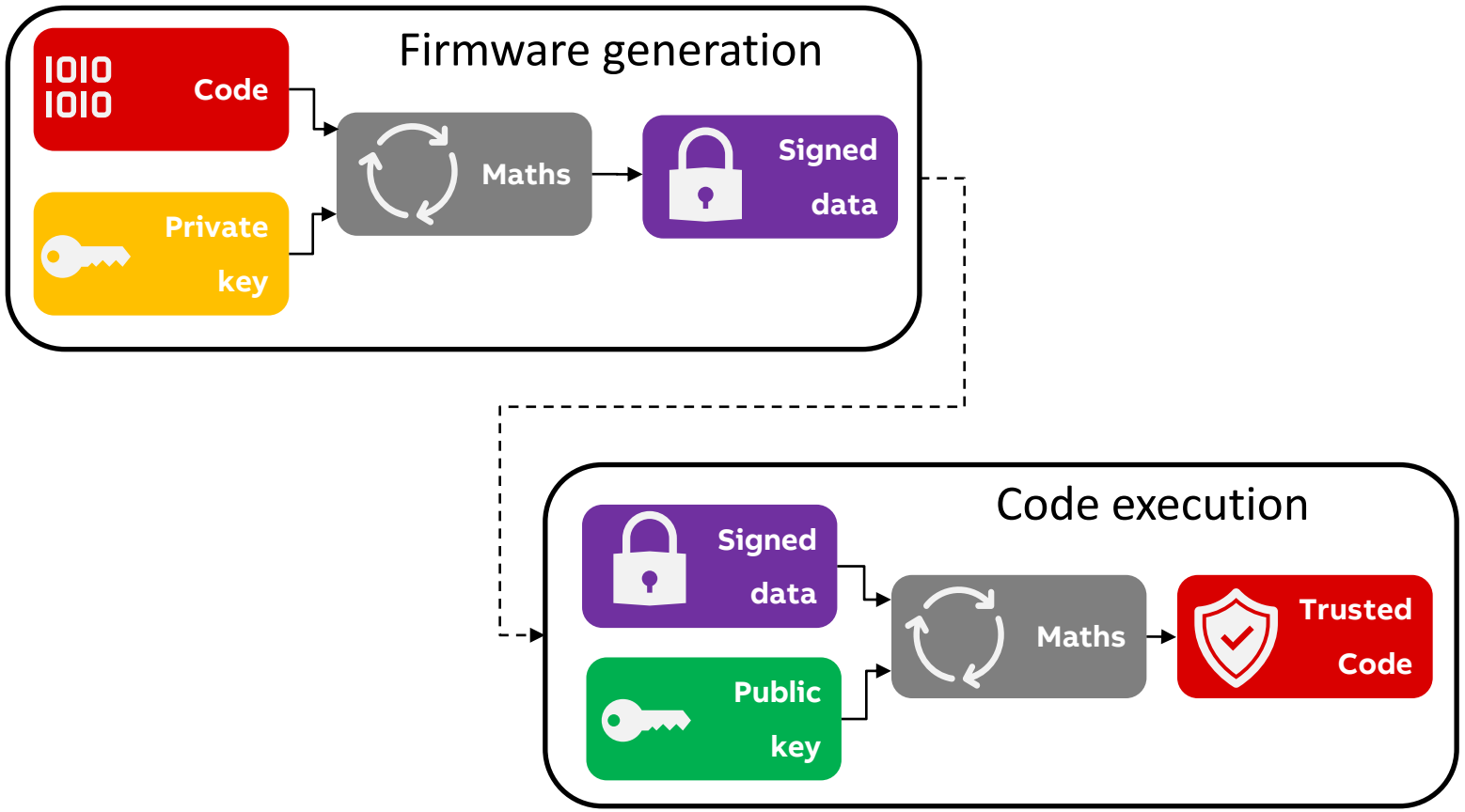


Hashing

Hashing uses mathematical function to map arbitrary size data to fixed-size string of text.

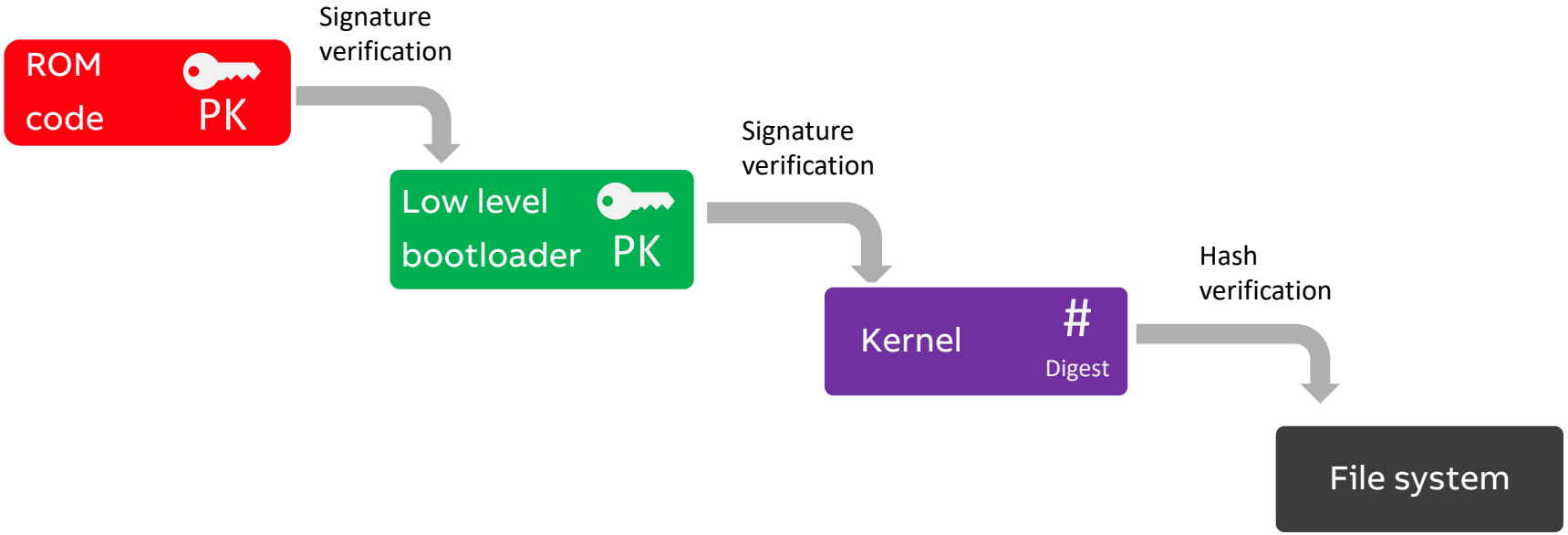


Code signature



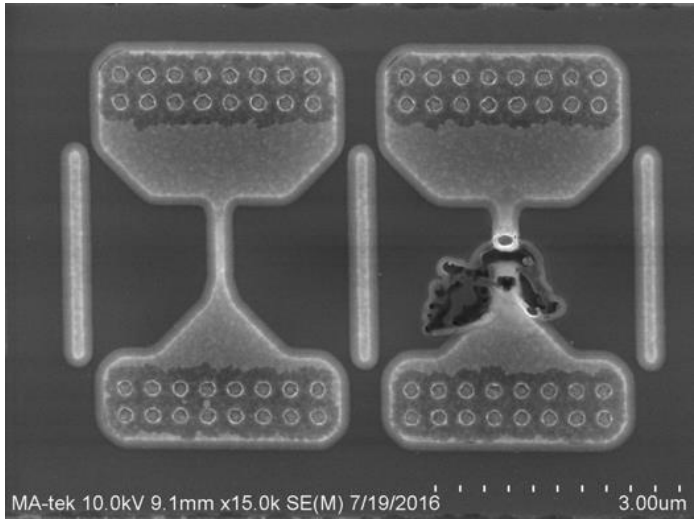
Secure boot

- Every component verified using its digital signature and public key
- Filesystem integrity verified using hash
- If any element is authenticated but not sufficiently lockdown (i.e. serial port/ssh access) chain of trust might fail



ROM code

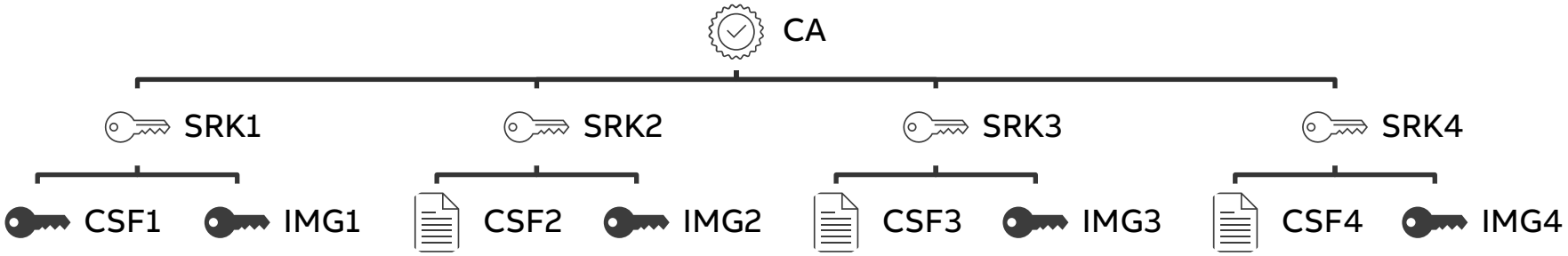
- The implementation of the very first boot stage is vendor-specific.
- Hardware support is required to establish an initial trust anchor.
- The foundation for the secure boot can be located in a dedicated module inside SoC or embedded during production into immutable non-volatile BootROM memory.
- On NXP iMX6, there is a hardware component called High Assurance Boot (HAB) that can validate the signature of the first stage bootloader.
- eFuses - One-Time-Programmable (OTP) fuses.



Programmed eFuse (Source: MA-Tek)

HAB

- Super Root Key (SRK) - RSA key pair verified at the boot-time by ROM code
- PKI can contain up to 4 SRKs
- Command Sequence File (CSF), a binary data structure interpreted by the HAB to guide authentication operations



HAB boot

- Loads bootloader to secure space
- Loads SRK from image
- Compare SRK hashes with OTP
- Check image is properly signed
- Execute bootloader binary
- Possible to load encrypted images

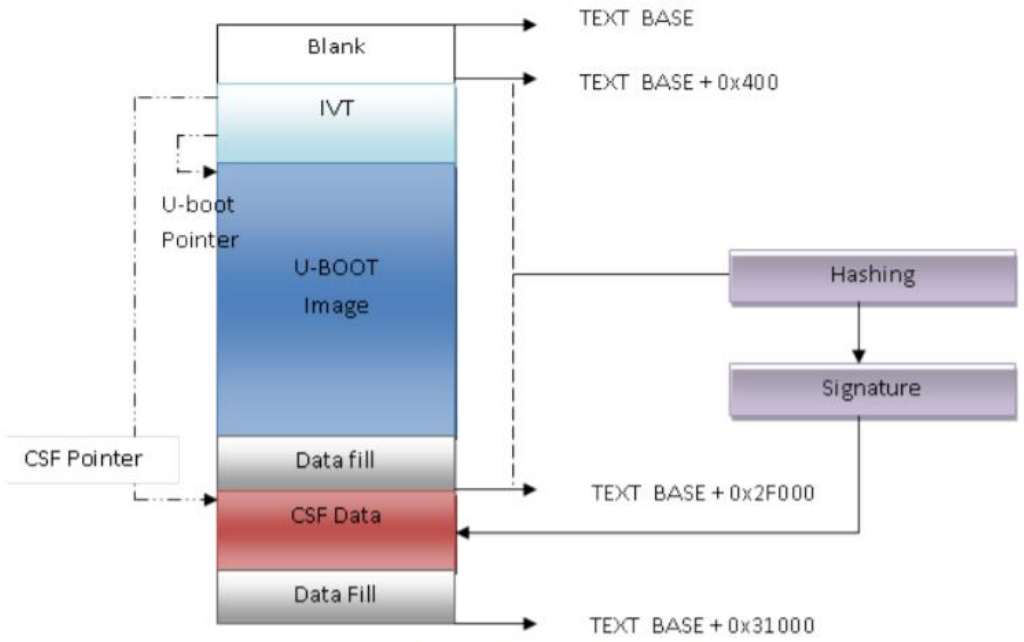
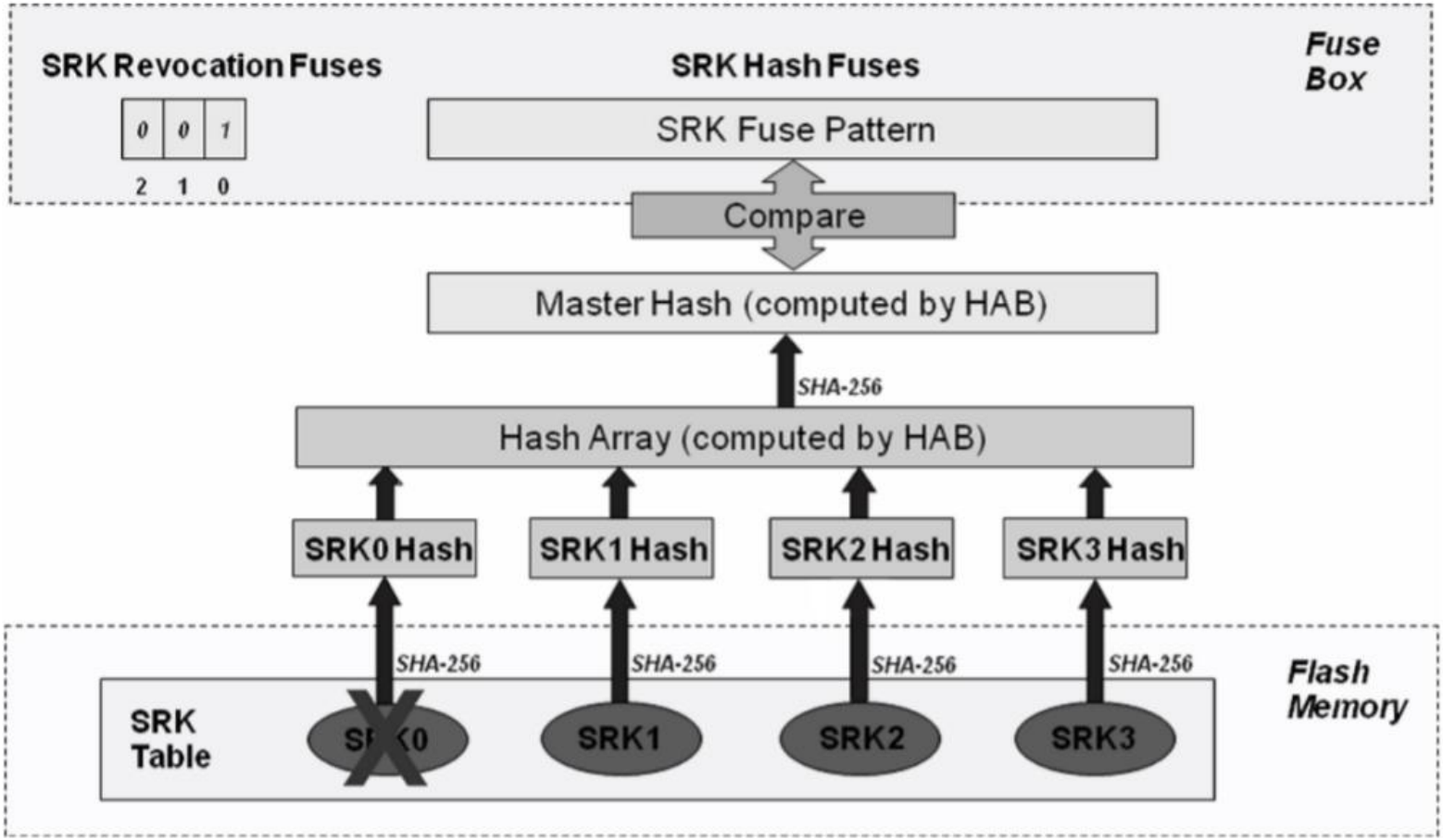


Figure 1: Structural Overview of Secure Boot Image

SRK revocation

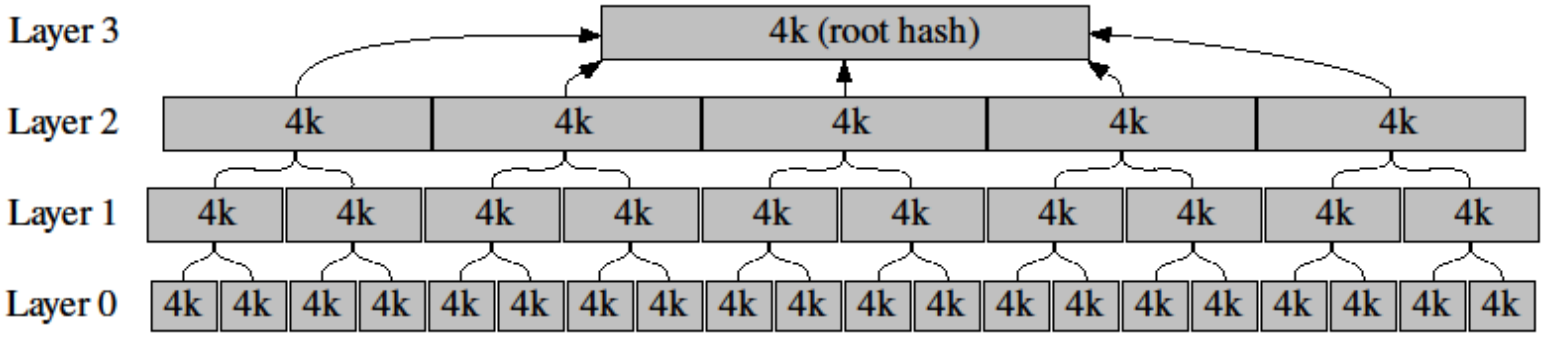


Low level bootloader

- U-Boot/Barebox
- Init hardware loads OS (first&second stage bootloader)
- FIT-image used to store kernel and other images.
- Device tree used to store public key of FIT-image.
- After boot verification device can be „closed”

Root file system

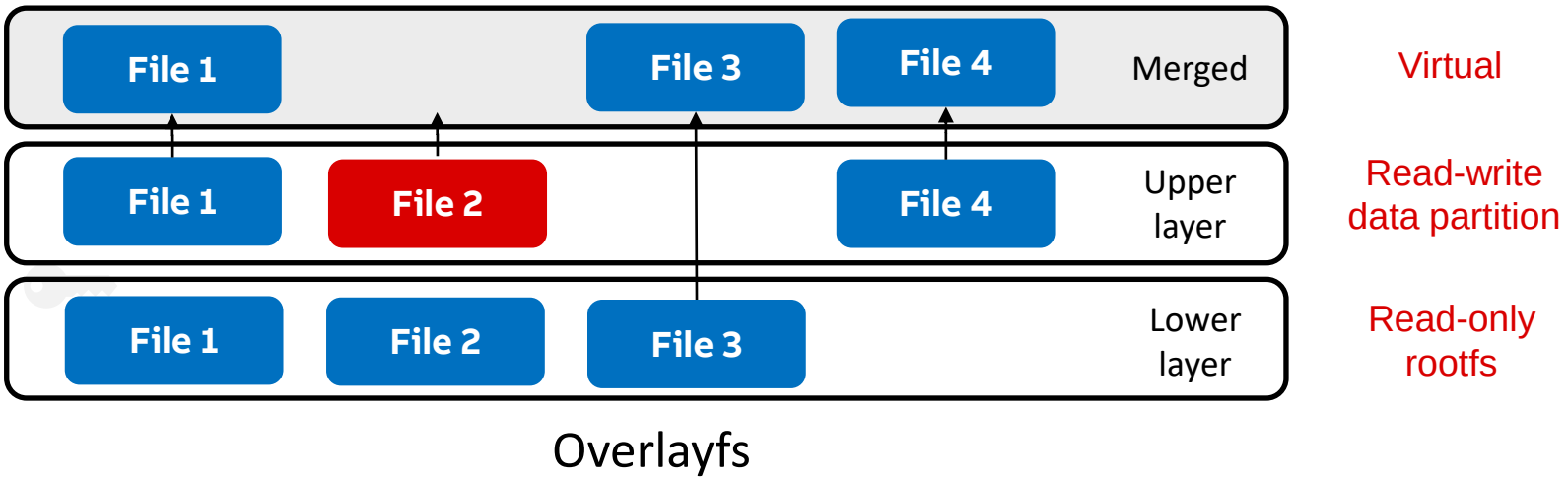
- Read-only vs. read-write filesystem
- dm-verity provides transparent integrity checking of read-only block devices.
- A hash tree is stored inside the root filesystem partition after the actual ext4
- Root hash is transfered to system via kernel command line (part of FIT-Image)
- All hashes will be verified on-demand during disk access



Markle tree of dm-verity

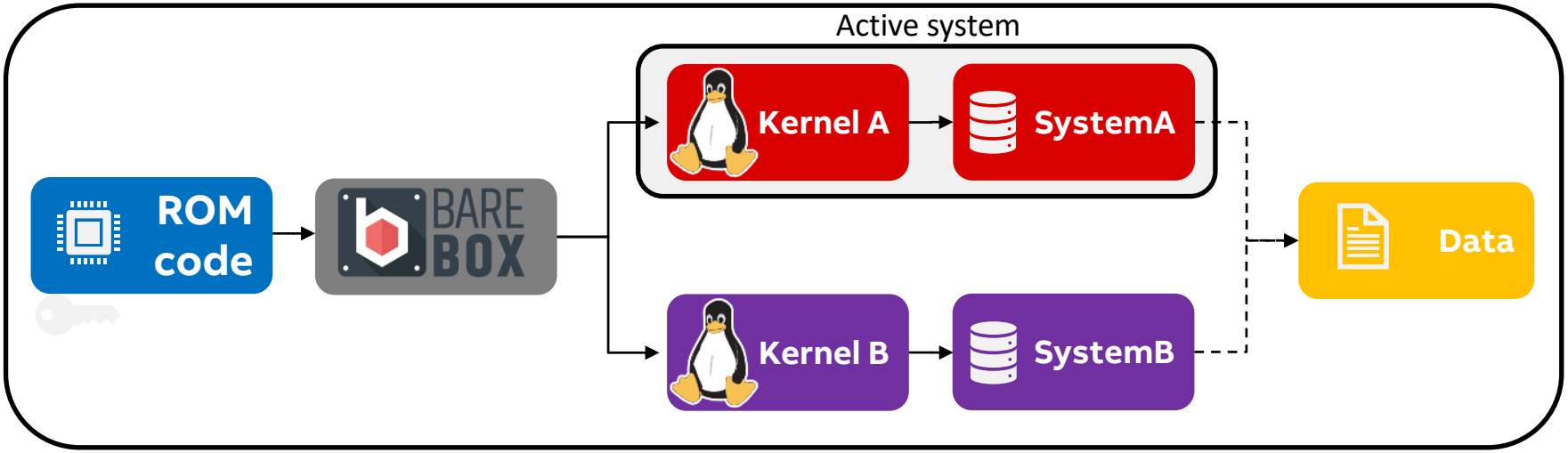
Root file system

- Systemd requires read-write /etc
- Need of place to store configuration
- Overlayfs over /etc
- List of allowed files + unstoppable systemd watching service to protect system



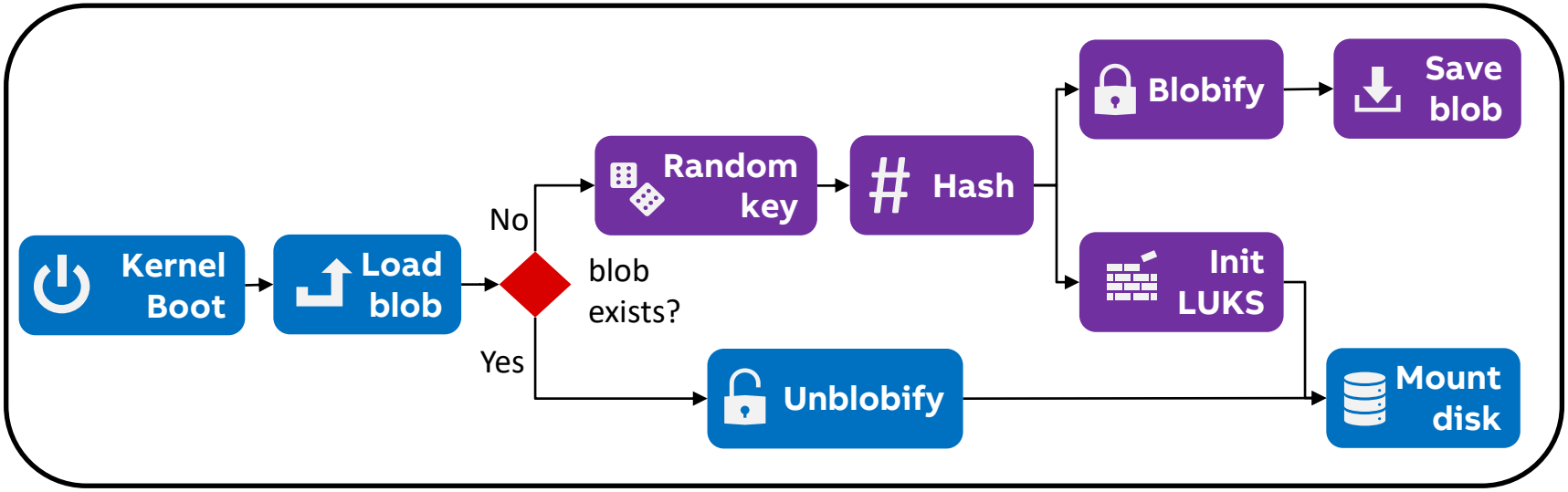
SystemA/SystemB

- Different scenarios SystemA/SystemB vs. Primary/Rescue
- Expected change during firmware upate
- Unexpected change by low-level bootloader after watchdog trigger (unsuccessful boot)
- State in lowlevel bootloader memory
- Shared data problem
- Infinite boot loop problem



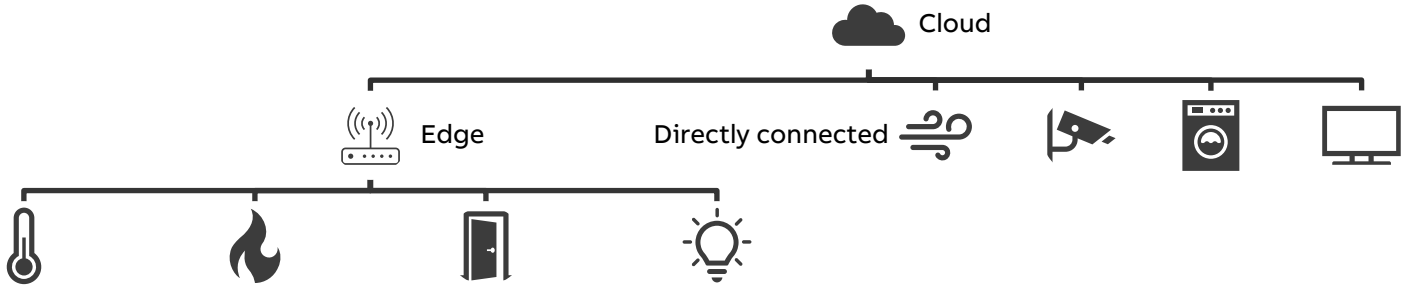
Encrypted storage

- Why to encrypt user data?
- dm-crypt + LUKS implementation
- How to protect the encryption key?
- Blobification using OTP Master Key
- Software vs. Hardware encryption
- Initialization problem



Secure communication

- Moving device logic/functionalities to the cloud increase the risks of attack
- Uninterrupted chain of trust
- Secure communication needed
- Proof device genuineness needed

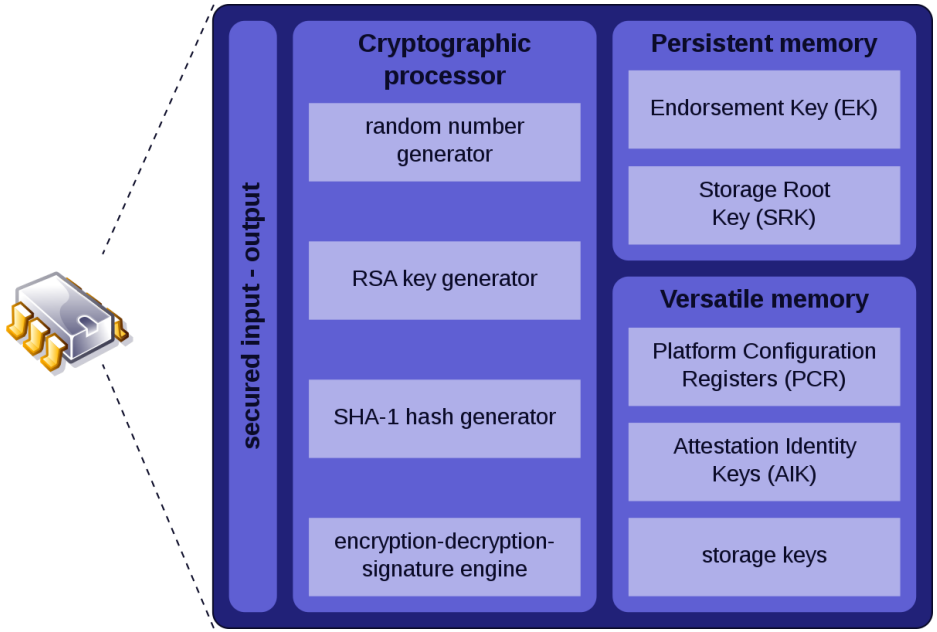


Security recommendations for secure communication

- Each device should have a unique public/private key pair
- Keep the private key secret
- Only secure protocols (e.g., SCP, SSH, TLS, IPSec, and HTTPS) shall be used by default
- Use secure provisioning process
- Prepare way to secure update device firmware
- Prepare way to update root certificates
- Key pairs should be rotated periodically if needed
- Ensure correct time on the device

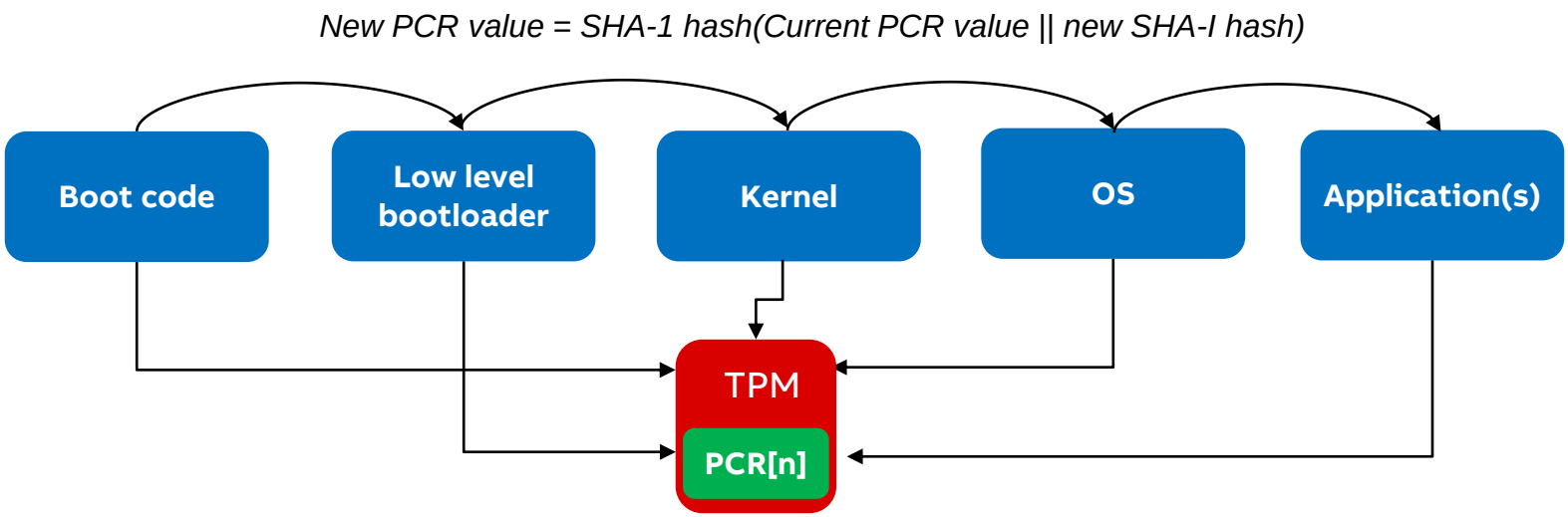
Trusted Platform Module (TPM)

- International standard well adopted by computer industry
- Small crypto engine
- Security by (hardware) separation
- Can be used for secure boot and secure storage



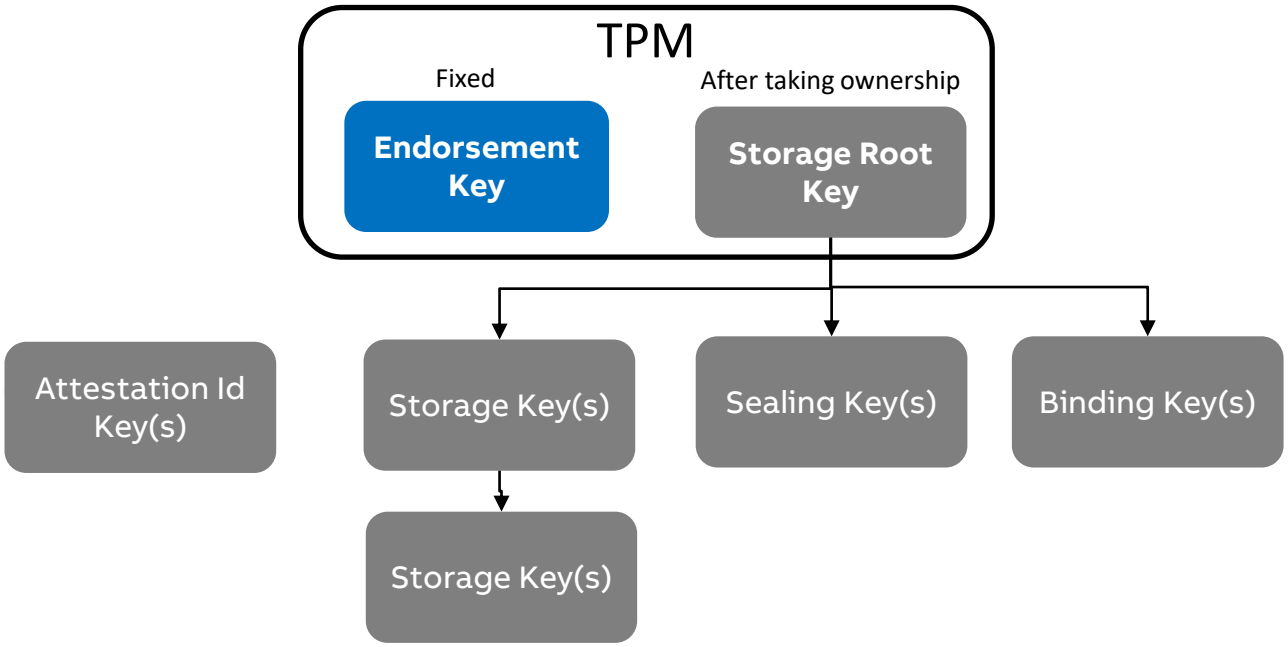
TPM Platform configuration registers

- Boot integrity measurement
- New value depends on current value
- Unlimited number of measurements
- Possible to readout PCR in trusted way



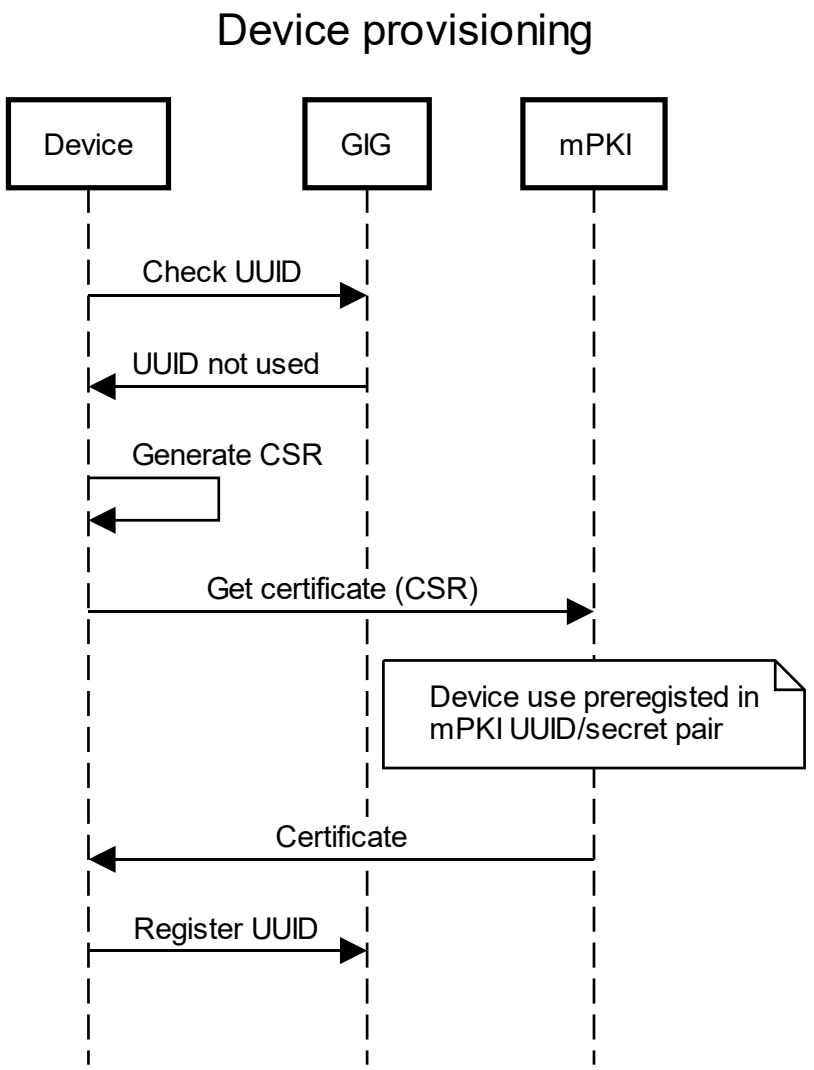
TPM key hierarchy

- EK unique for TPM, AIKs to prove genuity of TPM
- SRK is created when user takes ownership of TPM,
- Private key never leaves TPM
- Password as proof of ownership



Device provisioning

- UUIDv6 used as certificate common name
- Global ID Generator (GIG) server
- Managed Public Key Infrastructure (mPKI) Service
- Dedicated mPKI profile
- Unique one-time-use pair UUID/secret used to acquire certificate
- SCEP protocol used for communication with mPKI
- Modified SCEP client ensures the private key is sealed in TPM



Common Vulnerabilities and Exposures (CVE)

- Hard to prevent potential security threats if you are using Linux or other open-source software
- Need to monitor known vulnerabilities before after the release (CVEs databases)
- Process in place to deal with relevant 3rd party software security updates and patches
- Keep your Buildroot/Yocto/PTXDist/.. updated
- Firmware update path needed!

Principle of least privilege (PoLP)

- Entity must only have access the information or resources necessery to perform its function
- Split system functionalities to multiple users
- Implement a granular permission system
- Use mount options (ro, noexec, nosuid..)
- Minimal set of services/ports/software installed on the device
- No backdoor accounts and hardcoded credentials
- Be careful with wildcards in sudoers:

```
# ledsandreset user can read KNX port stats to calculate LED blink time
Cmnd_Alias TTY_STATS_READ = /usr/bin/cat /proc/tty/driver/*
ledsandreset ALL= NOPASSWD:NOLOG_INPUT:NOLOG_OUTPUT: TTY_STATS_READ

# Can read every file on rootfs
sudo –u ledsandreset ls /usr/bin/cat /proc/tty/driver/../../../etc/sudoers
```

Others

Static (and dynamic) code analysis	Can be used to detect potential attack vectors like: null pointer de-references, freeing already freed memory, overflowing fixed size buffer and many others
Compiler warnings	Warnings generated during compilation, package installation (npm/NuGet/pip/bower) or minification/bundling (JavaScript, HTML, CSS).
Code reviews	Code reviews improve code quality, help programmers build relationships and work together.

Testing

- When it comes to security, one of the most important elements of software process is testing
- In addition to standard tests (unit tests/integration testis/system tests/. . .), it is highly recommended to perform dedicated security tests
- Security tests often mimics the actions performed by hackers
- Fuzz testing involves automatic tools to input massive amounts of random and pseudorandom data. In case of Ethernet devices this might be storm of valid and invalid packages. Tools for fuzz test: OpenVAS, OWASP ZAP, NMAP
- Penetration test is an authorized simulated cyberattack on a computer system, performed to evaluate the security of the system. It is manual process and requires more knowledge

Credits

- Pengutronix team
- Bootlin team
- ABB team
- OSS software community
- Online materials:
  - Securing Embedded Linux Systems with TPM 2.0 - Philip Tricca, Intel
  - Secure Boot from A to Z - Quentin Schulz & Mylène Josserand, Bootlin
  - Conceptual Design and Implementation of a Secure Bootchain based on the High Assurance Boot (HABv4) Architecture of the NXP platform, Friedemann Lipphardt, Bachelor Thesis