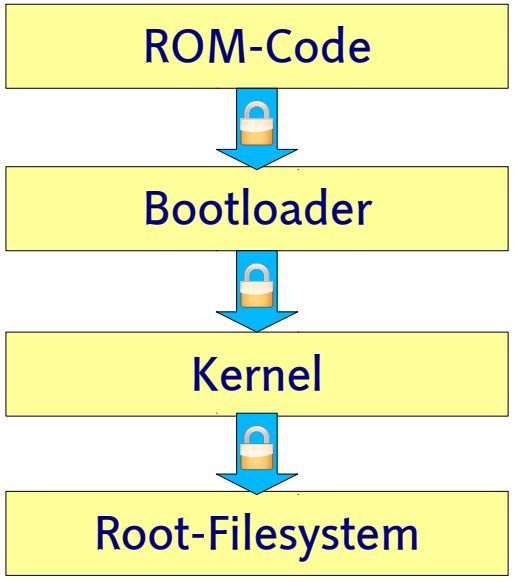
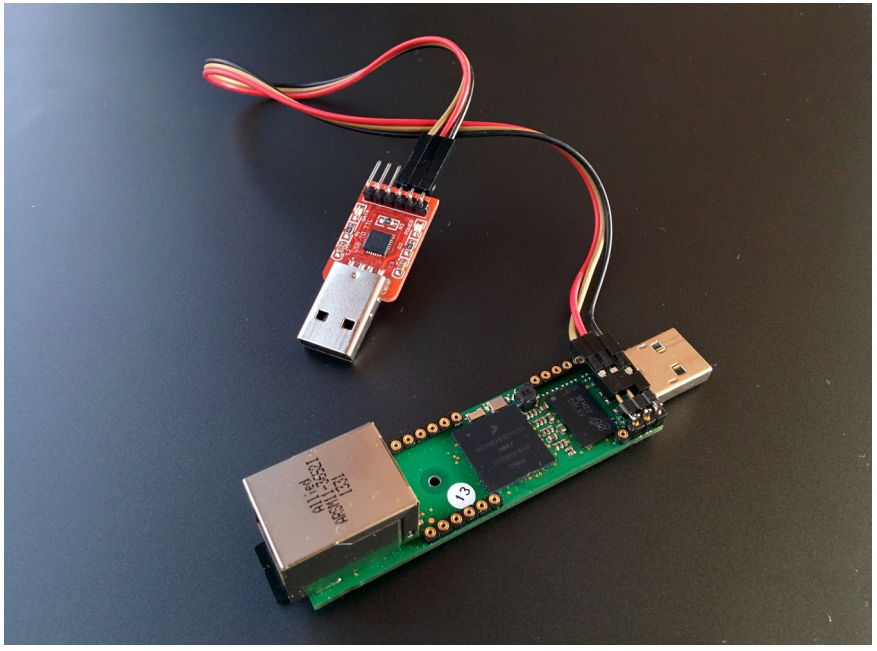
Verified Boot mit Linux auf ARM





Chemnitz, 21.3.2015

Jan Lübbe <jlu@pengutronix.de>

Pengutronix.

Jan Lübbe

- Linux seit 1998, Debian Developer
- Informatikstudium an der TU Braunschweig, Dipl.-Inform.
- 2008: Freelance-Linux-Entwicklung
  (OpenMoko, Mobilfunk-Infrastruktur & Osmocom, ...)
- Seit 2012 bei Pengutronix:
  Systemarchitekt im Kernel & Platform Team

Agenda

- Verified Boot?
- Bootloader
- Kernel
- Root-Dateisystem
- Wie geht's weiter?

Zwischenfragen sind erwünscht!

Warum Verified Boot?

Attraktives Angriffsziel:

- Linuxsysteme steuern kritische Industrieprozesse
- Eingebettete Systeme werden im Vergleich zu Servern schlechter gewartet

Komplexe Software:

- Jedes Linux-System hat unentdeckte Sicherheitslücken
- Kommerzielle Steuerungssoftware
- Defense-in-Depth ist wichtig!

Wir können es selbst:

- SoCs mit Hareware-Support sind überall zu kaufen
- Software-Support als FOSS

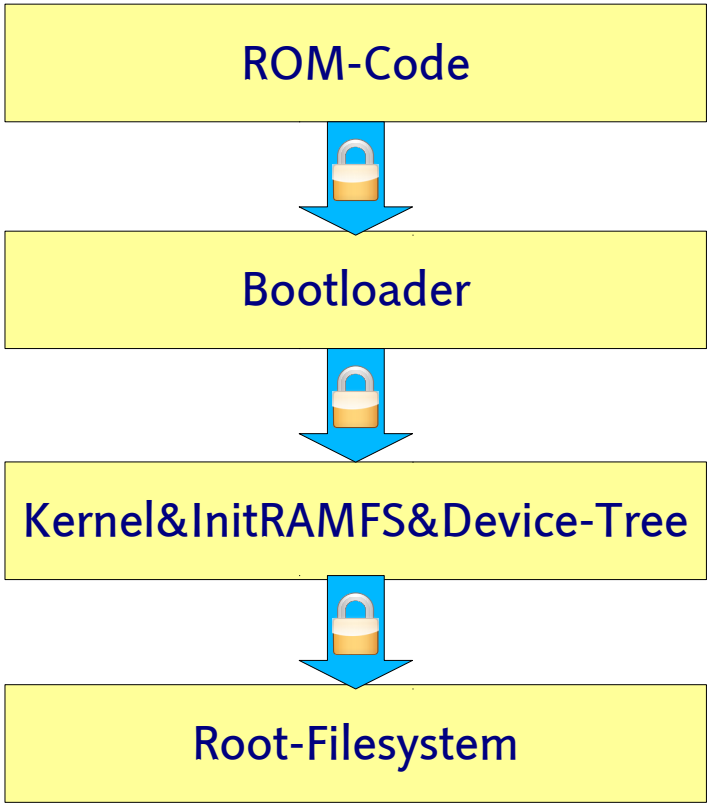
Pengutronix.

Was wollen wir schützen?

- Bootloader
- Kernel
- Dateisystem
  - Programme
  - Konfigurationsdaten
  - Anwendungsdaten

- Der Angreifer kann alle gespeicherten Daten manipulieren
  → wir wollen jede Manipulation erkennen

Bootvorgang



Herstellerabhängig (hier Freescale HABv4, SHA und RSA)

FIT-Image (SHA und RSA)

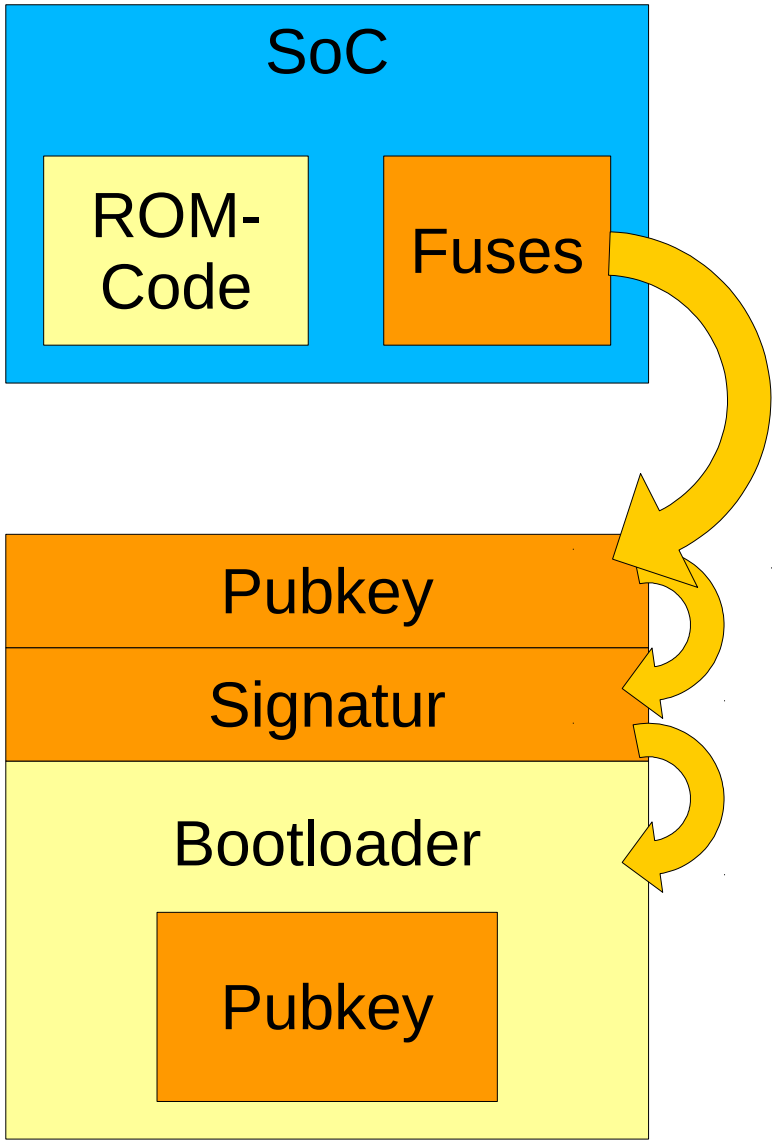
IMA & EVM (HMAC und RSA)

Pengutronix.

Bootloader

- Üblicherweise auf ungeschütztem Speicher (NAND, eMMC, SD)
- Hat volle Kontrolle über das System
- Muss vom ROM-Code geprüft werden
  - Hash des Zertifikats wird in On-Chip-Fuses gebrannt
- Enhält den Public-Key zur Prüfung des Kernel-Images

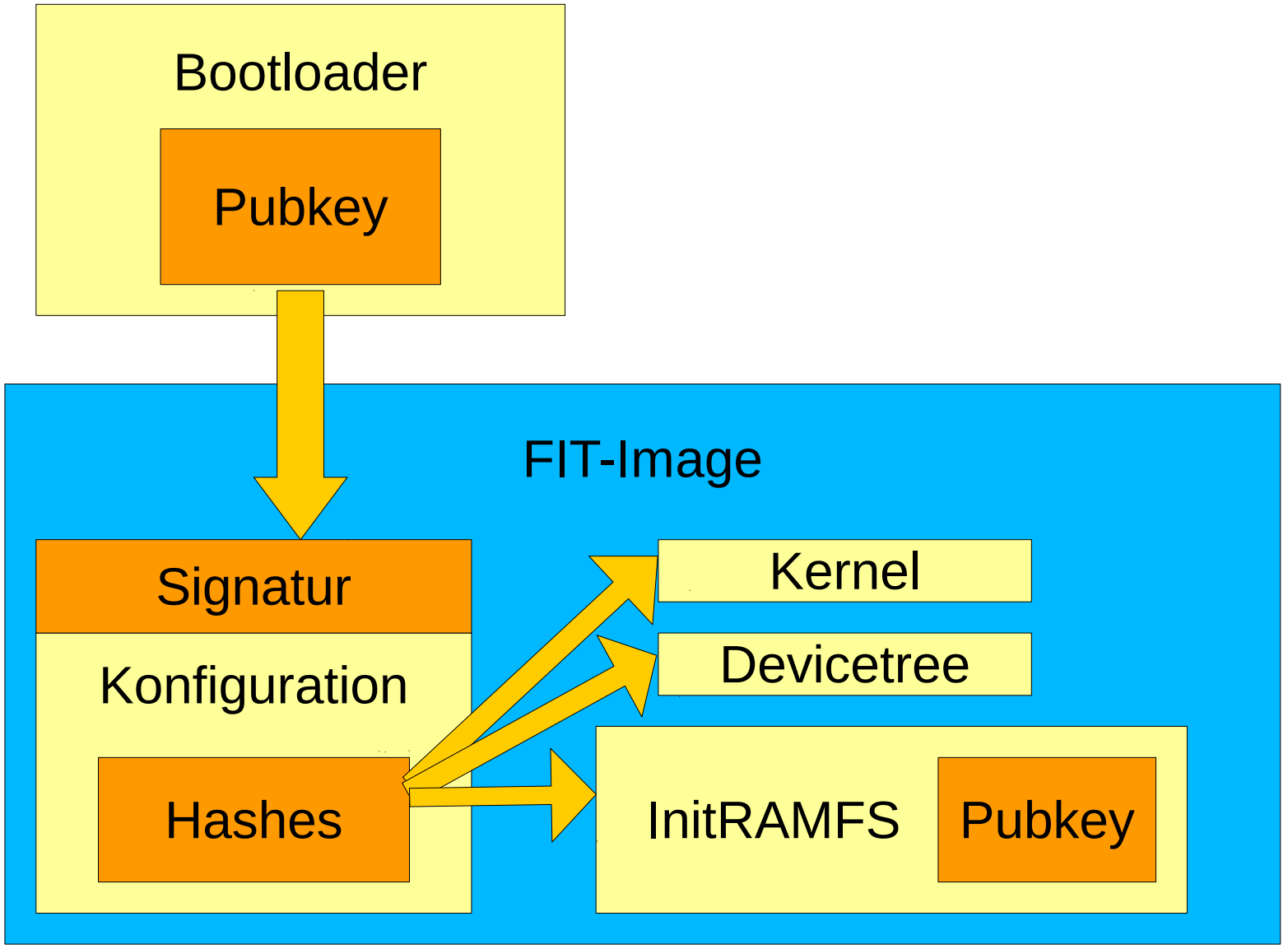
Bootloader



FIT-Image

- In eigener Partition oder im Root-Dateisystem
- Enthält Kernel, InitRAMFS und Device-Tree
  - Kann auch mehrere Varianten enthalten
  - Signiert ist immer eine „Konfiguration“ aus Kernel, InitRAMFS und DT um Mix-and-Match-Angriffe zu verhindern
- Muss vom Bootloader geprüft werden
  - Signatur passt zum Public-Key im Bootloader
- Enhält den Public-Key zur Prüfung des Root-Dateisystems

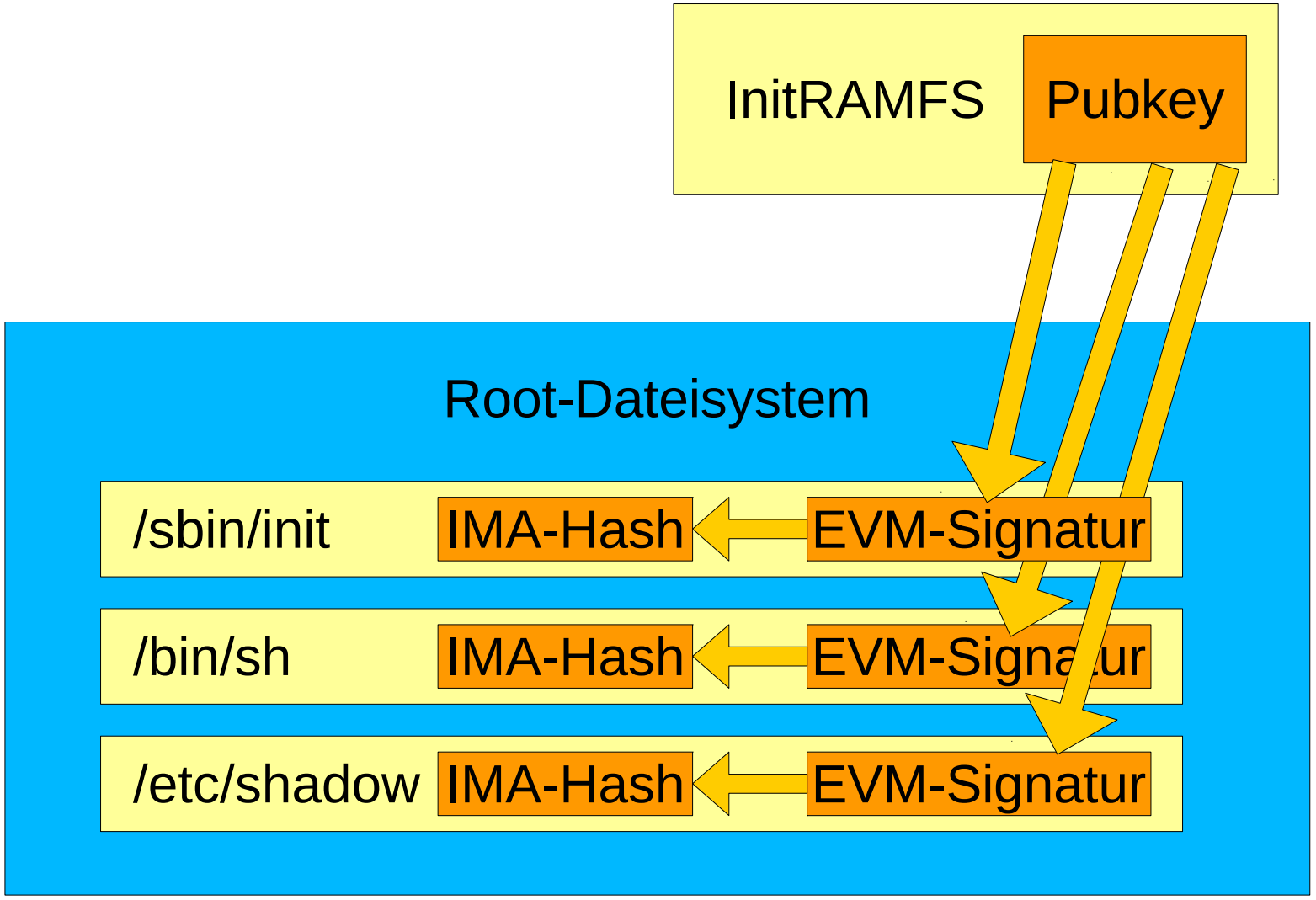
FIT-Image


Bootloader
Pubkey
FIT-Image
Signatur
Konfiguration
Hashes
Kernel
Devicetree
InitRAMFS
Pubkey

Root-Dateisystem (initial)

- ext4 oder UBIFS
  - Extended Attributes
- Jede Datei hat einen IMA-Hash
  - SHA1 oder SHA256 über den Datei-Inhalt
- Jede Datei hat eine EVM-Signatur
  - Wird auf dem Entwicklungsrechner mit einem Private-Key signiert
  - RSA-Signatur passt zum Public-Key im InitRAMFS

Root-Dateisystem (initial)



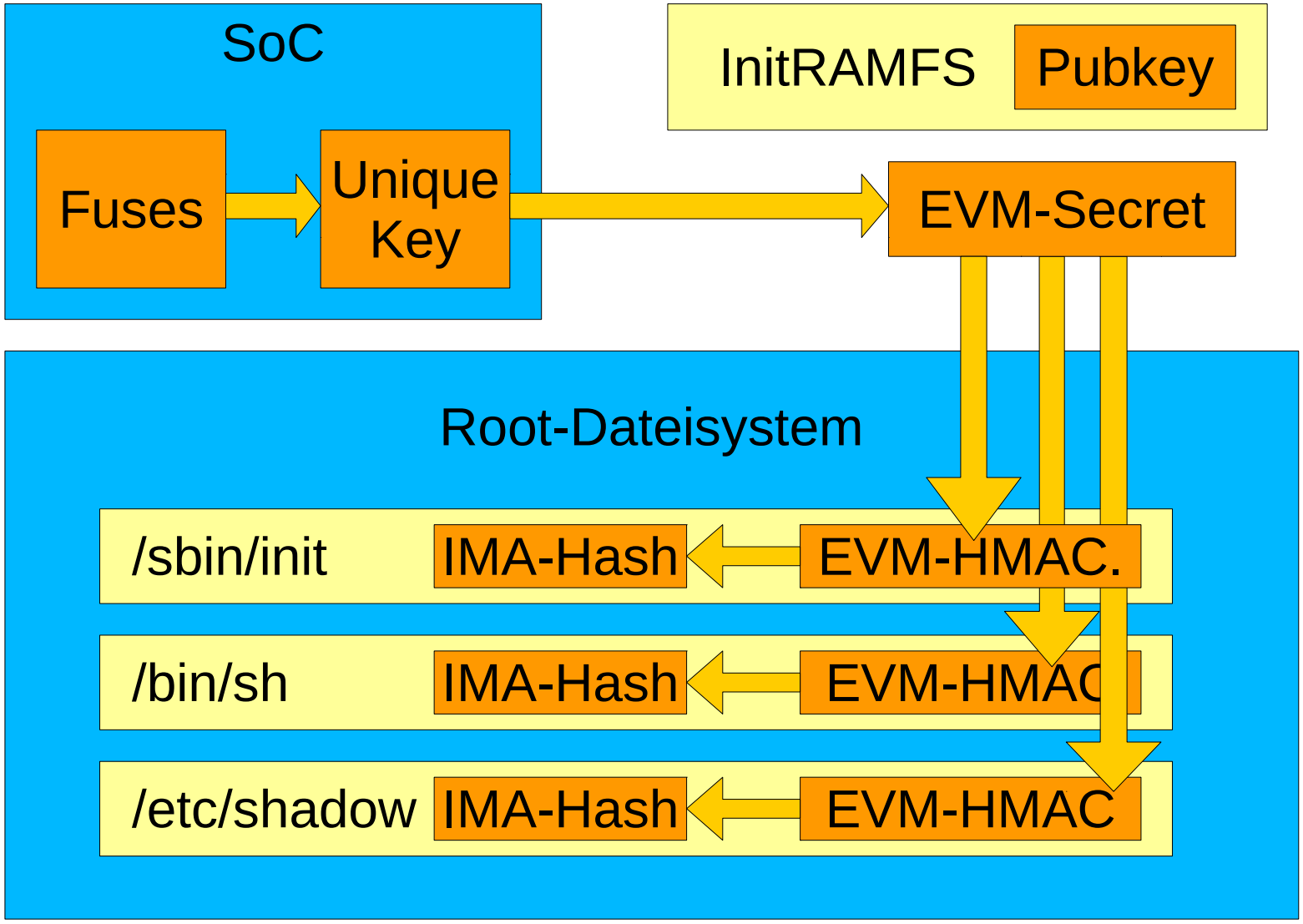
Pengutronix.

Root-Dateisystem (schreibbar)

- Keine RSA-Signaturen
  - Wir haben keinen Private-Key auf dem System
  - RSA ist langsam
- stattdessen SHA-HMAC
  - Benötigt ein Secret, dass für jedes System unterschiedlich ist
  - Beim ersten Zugriff wird die Signatur durch den HMAC ersetzt

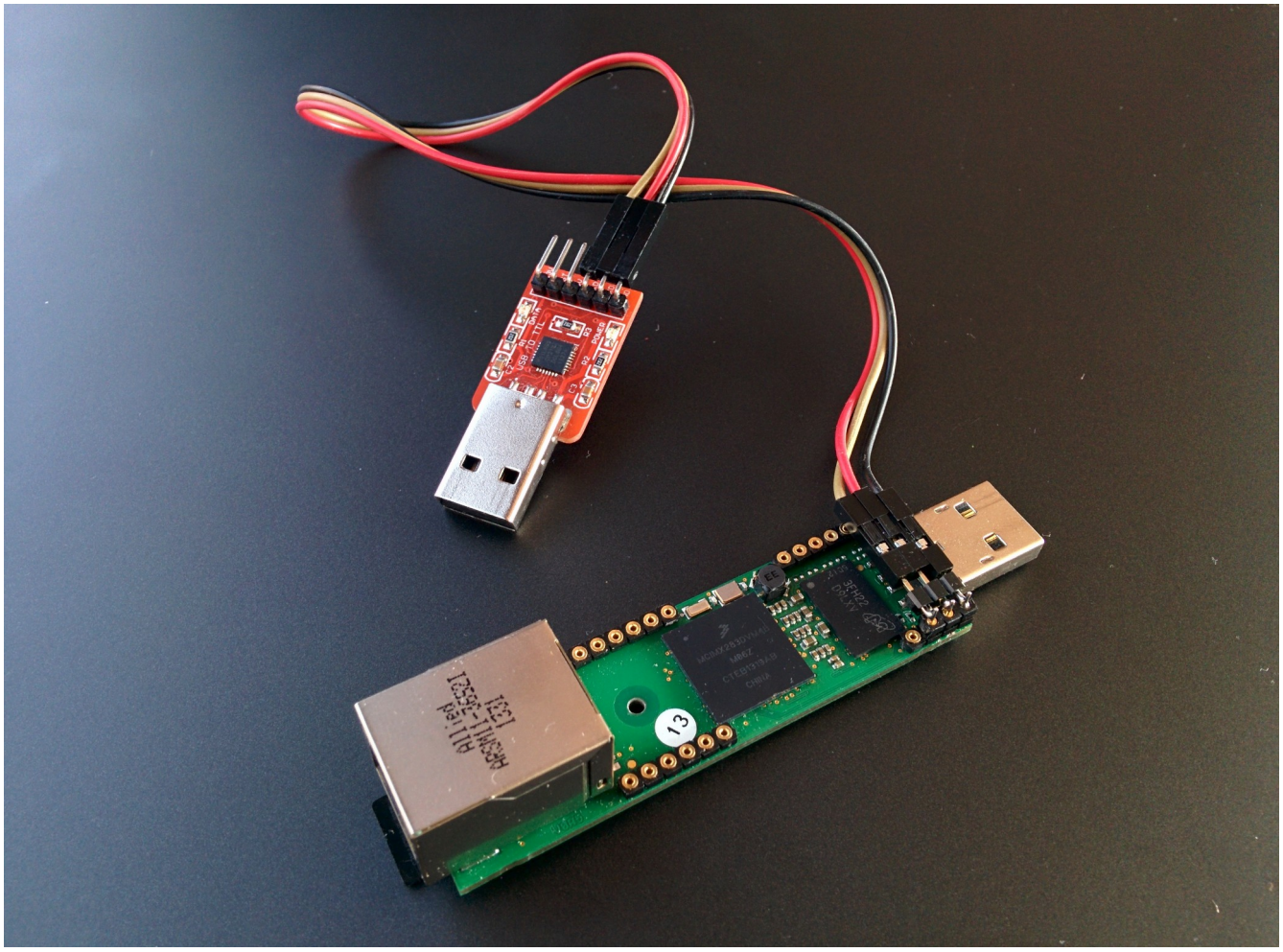
- Jede Datei hat eine IMA-Hash und EVM-HMAC
  - Nur ein korrekt gebootetes System hat Zugriff auf das EVM-Secret
  - Angreifer können für manipulierte Dateien keinen passenden HMAC berechnen

Root-Dateisystem (schreibbar)



Pengutronix.

Demo



Selber machen!

- Freescale MX28
  - I2SE Duckbill
  - Olinuxino?
- Freescale MX53
  - USB Armory
- Freescale MX6
  - Cubox-i
  - RioT-Board

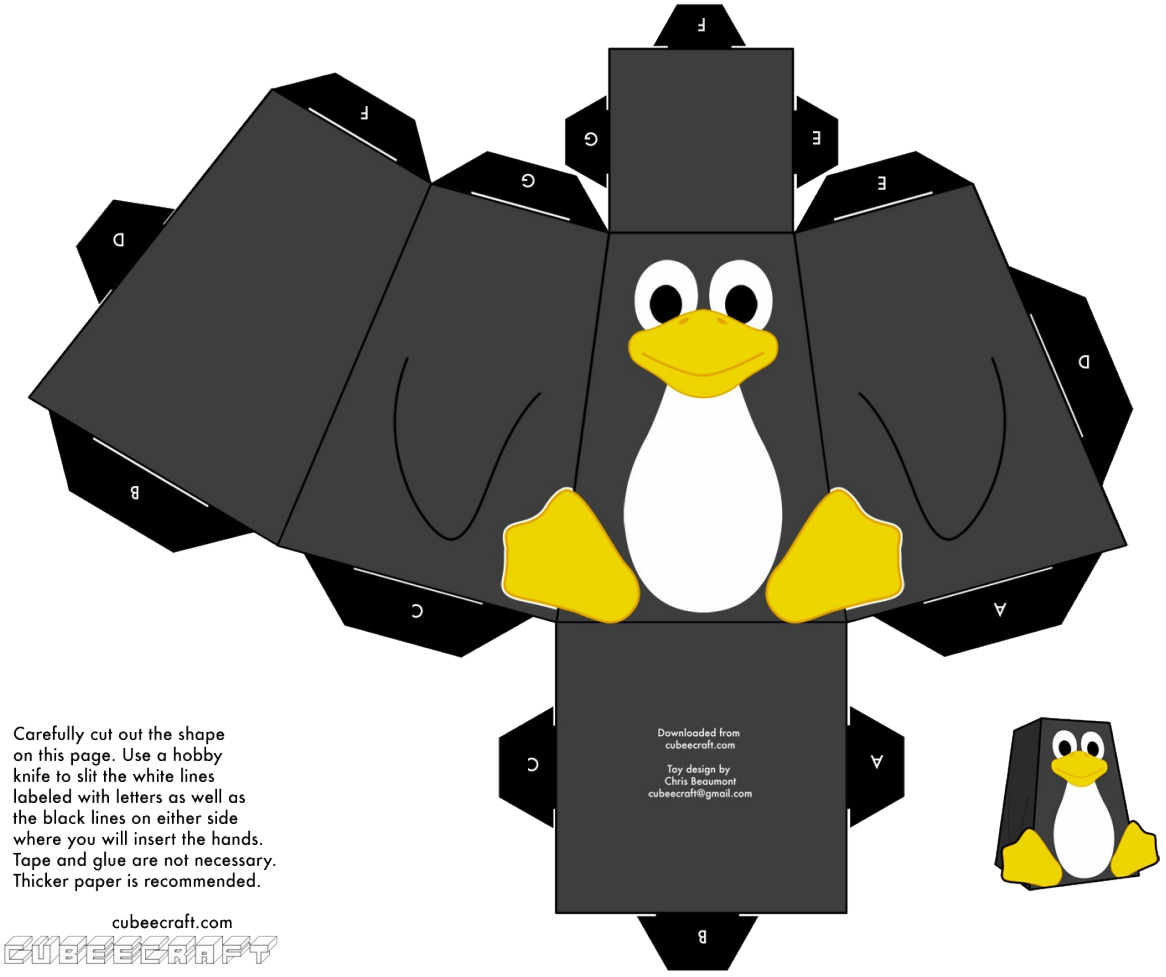
- Ohne Hardware-Support: Read-Only SPI-NOR oder eMMC + TPM

- Beispiel-BSP für MX28-Duckbill wird demnächst veröffentlicht

Was fehlt noch?

- Schutz von Verzeichnissen
  - Verhindert Verschieben, Löschen und Anlegen von Dateien
  - Es gibt schon Patches „directory integrity protection“
- Integration in mkfs.ext4 und mkfs.ubifs
  - Teilweise gibt es Patches, aber Details sind noch ungelöst
- Verschlüsselung von Anwendungsdaten mit einem Device-Key per ECryptFS
- Unterstützung für mehr SoCs:
  - MX53 (Zugriff auf Device-Key)
  - MX6 (Crypto-Einheit CAAM)
  - Andere Hersteller (Dokumentation?)

Q & A



Carefully cut out the shape on this page. Use a hobby knife to slit the white lines labeled with letters as well as the black lines on either side where you will insert the hands. Tape and glue are not necessary. Thicker paper is recommended.

cubeecraft.com

Pengutronix.