

PUNCH  BOOT

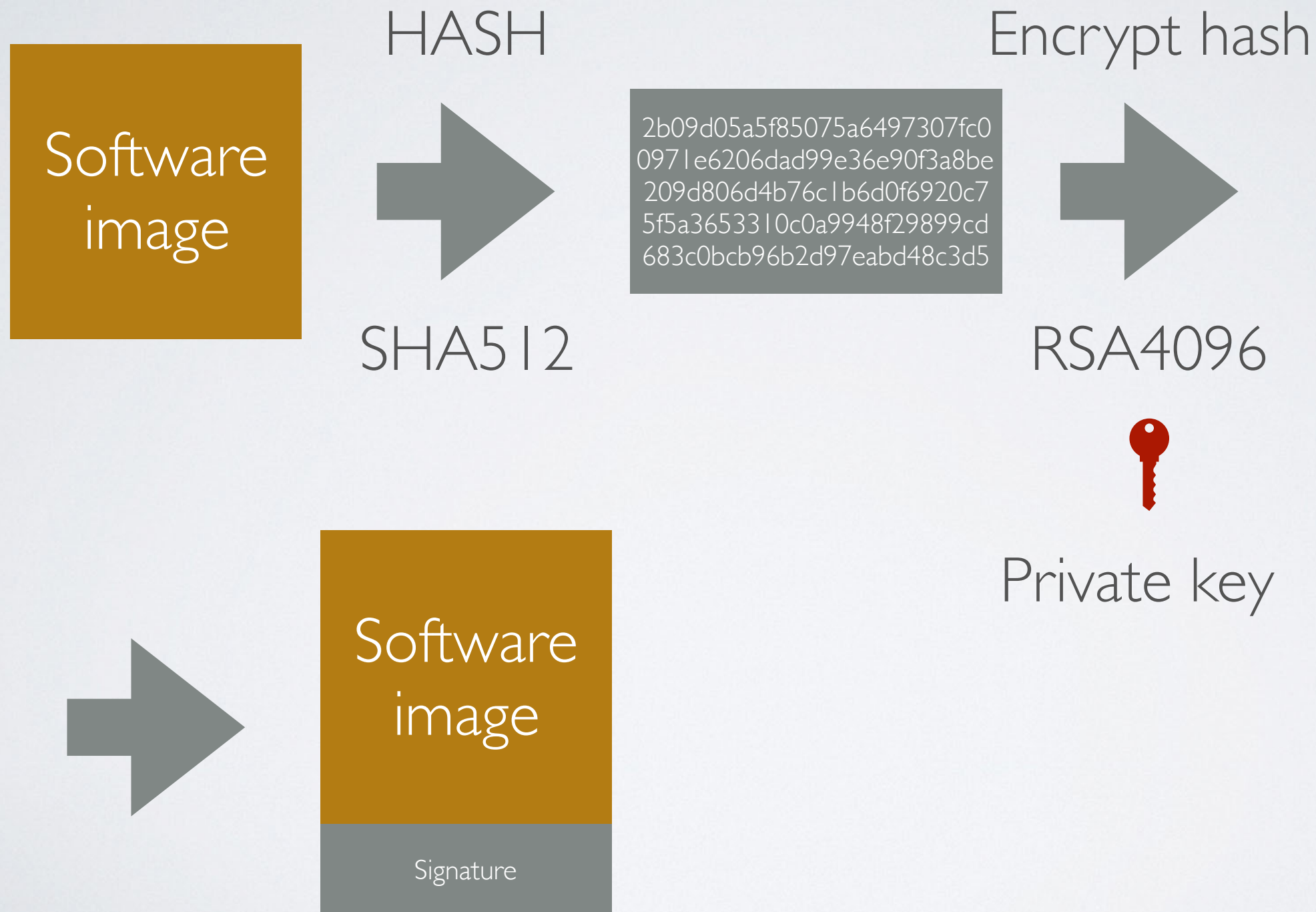
INTRODUCTION

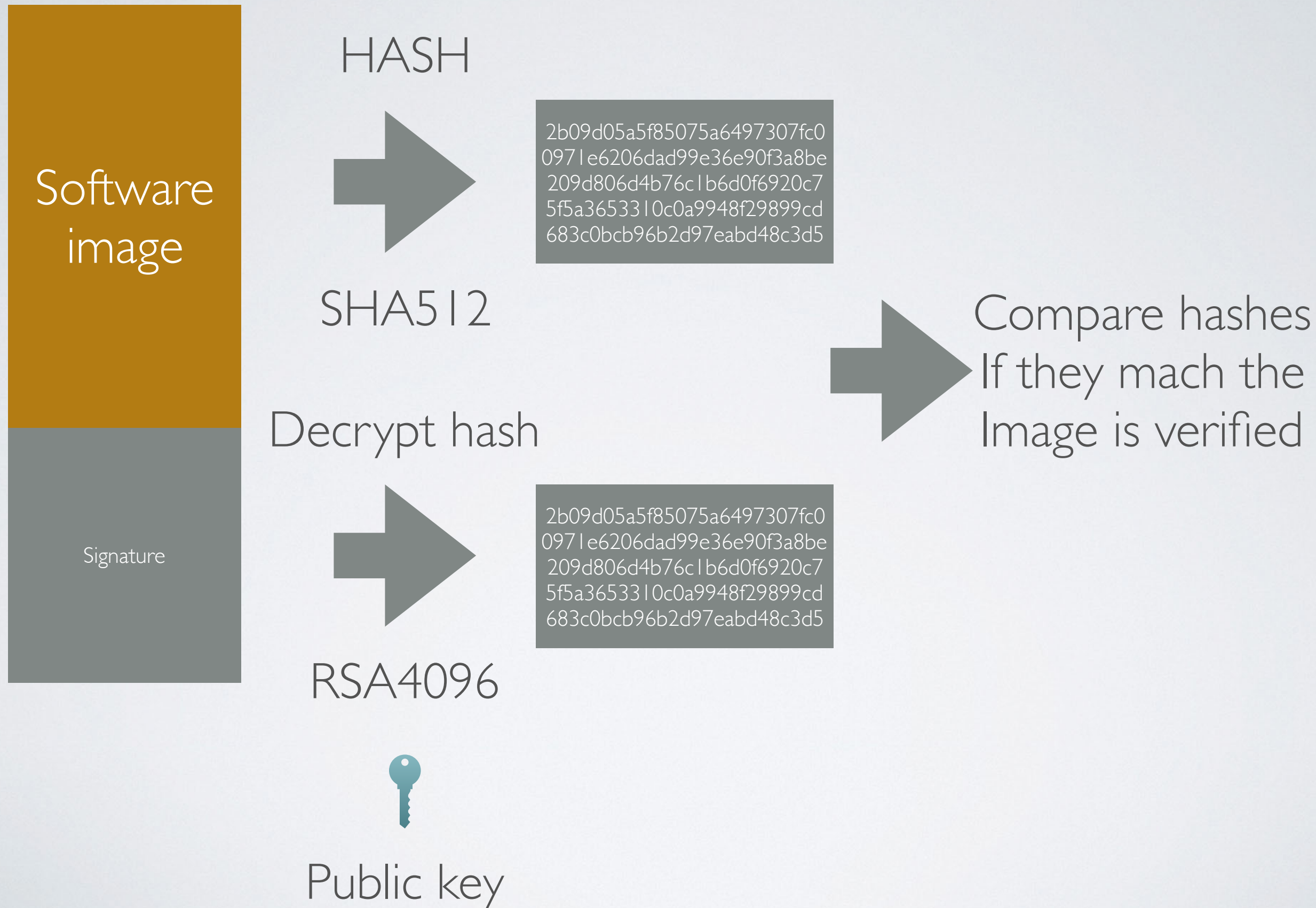
- Boot loader for embedded systems
 - No run time configuration
 - Minimalistic design
 - Good test coverage
- Focus on security and boot time while still being useable for day-to-day development
- Production software download
 - USB HS transfer speeds of 20 MBytes/s
 - Complete unit provisioning using only USB
- Software update primitives
 - A / B system switching to support atomic updates
 - Rollback

SECURE BOOT - BASICS

- Why secure boot?
 - Prevent malicious software from running
 - Supply chain integrity

CRYPTOGRAPHIC SIGNATURE





ROOT OF TRUST

- Public keys used for image verification must be fused into the CPU
- Size of the keys are impractical to store in OTP fuses due to size
- Hash of public keys are stored in OTP fuses which can not be changed
- Every boot the mask rom compares stored public keys hash to the stored OTP hash



WHAT PROBLEMS CAN PUNCHBOOT SOLVE

- Secure boot
 - Load and authenticate next software image
 - Cryptographic accelerators for computing hash'es and signatures
 - One hash and one signature for the complete image which might contain several images
- Production software download
 - Recovery mode allows high speed USB transfers which saves time in software download cell
 - Directly download boot loader image, kernel image and root filesystems
- Day-to-day development
 - Recovery mode can load images into RAM and execute them

PROBLEMS PUNCH BOOT DOES **NOT** SOLVE

- Punchboot does not secure the entire device
 - Root filesystem integrity is not covered
 - There are many ways to make a device insecure even if the boot loader is secure
- It does not provide a complete strategy for key management
- Punchboot is not tested in battle (yet)

DESIGN

- C99
- Supports ARMv7a and ARMv8 architectures
- GUID Partition Table (GPT) support
- Uses crypto hardware for fast hashing and signature verification (CAAM on imx)
- A/B atomic updates and rollback support
- Platform support for IMX6UL, IMX8M, IMX8X
- Released under BSD - 3

Code organisation - overview

- src/arch

- Armv8

- src/plat

- Imx8x

- src/board

- Imx8qxpmek

Low-level architecture,
Boot entry code,
Cpu init

Platform specific code,
Drivers
API: include/pb/plat.h

Board specific code,
Static configuration,
Fuse map,
GPT partition table,
Customisation hooks
API: include/pb/board.h

Building and common build options

make BOARD=imx8qxpmev LOGLEVEL=<n>

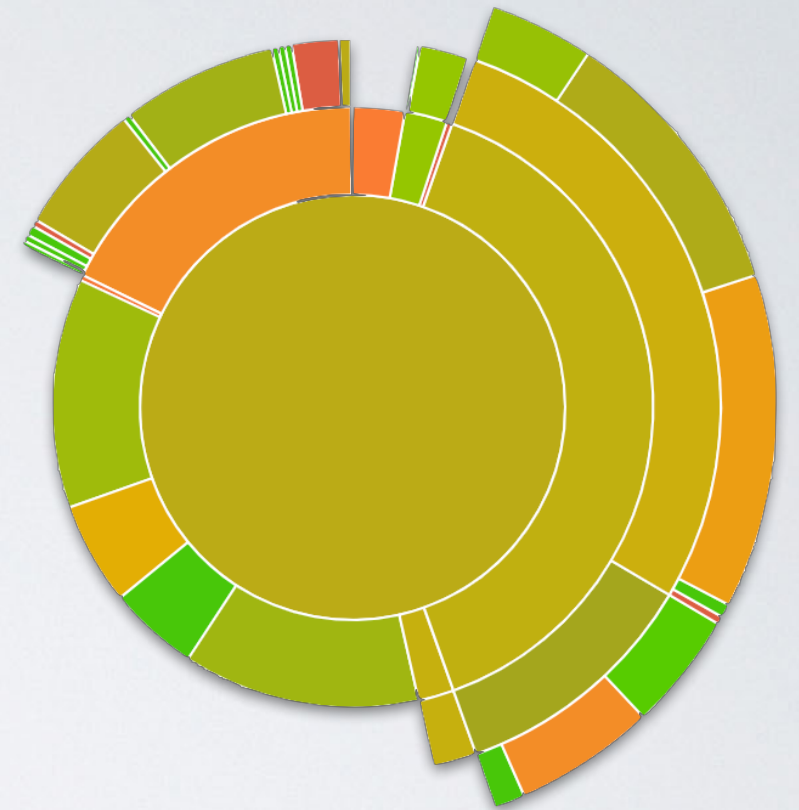


- CROSS_COMPILER - for specifying toolchain
- TIMING_REPORT - Disables all uart output but as a last step before jumping to the image entry point it will print a timing report for the different steps in the boot process
- BOARD_DIR=<...> (v0.3)
- Special build target 'test' to invoke test suite

0: No uart output
1: Errors only
2: Info and errors
3: Verbose

MODULE AND INTEGRATION TESTS

- Test suite runs in QEMU
- 85 % coverage
- Integration tests also cover support tools
- Static code analysis performed with synopsys coverity



Recovery mode

- If there is no active boot partition
- If the active partition is corrupt
- External event
- Recovery mode will automatically reset the device after a specified amount of time if USB does not enumerate (v0.2)
- Authentication cookie (in v0.2 or v0.3)

PUNCHBOOT CLI

- Supports different communication backends
 - USB
 - Domain socket (for testing)
- Can easily be integrated into other tools
- Punchboot library for integrating into other Tools and environments (v0.3)

--- Punch BOOT 3c0e ---

Bootloader:

- | | |
|---------------------------------------|------------------------------------|
| punchboot boot -w -f <fn> | - Install bootloader |
| punchboot boot -r | - Reset device |
| punchboot boot -b -s A or B | - BOOT System A or B |
| punchboot boot -x -f <fn> [-s A or B] | - Load image to RAM and execute it |
| punchboot boot -a -s A, B or none | - Activate system partition |

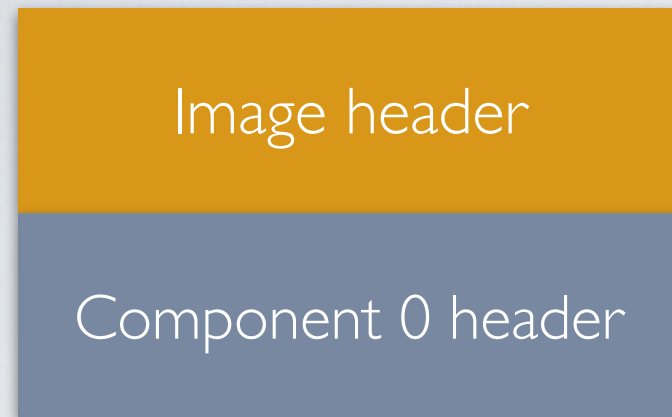
Device:

- | | |
|---------------------------------|------------------------------|
| punchboot dev -l | - Display device information |
| punchboot dev -i [-f <fn>] [-y] | - Perform device setup |
| punchboot dev -w [-y] | - Lock device setup |

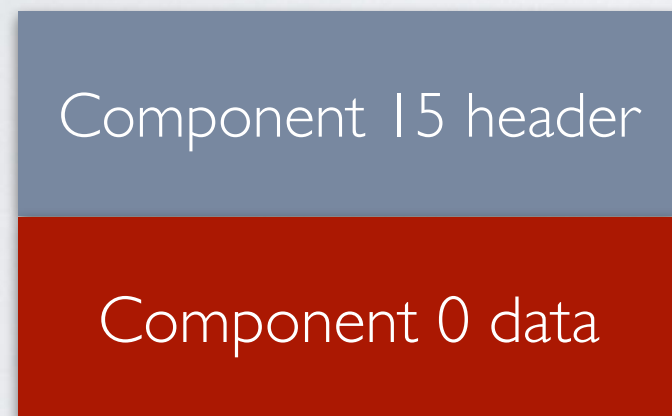
Partition Management:

- | | |
|----------------------------------|-------------------------------|
| punchboot part -l | - List partitions |
| punchboot part -w -n <n> -f <fn> | - Write 'fn' to partition 'n' |
| punchboot part -i | |

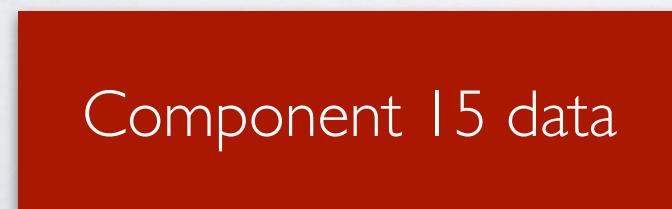
PUNCHBOOT IMAGE (PBI)



⋮



⋮

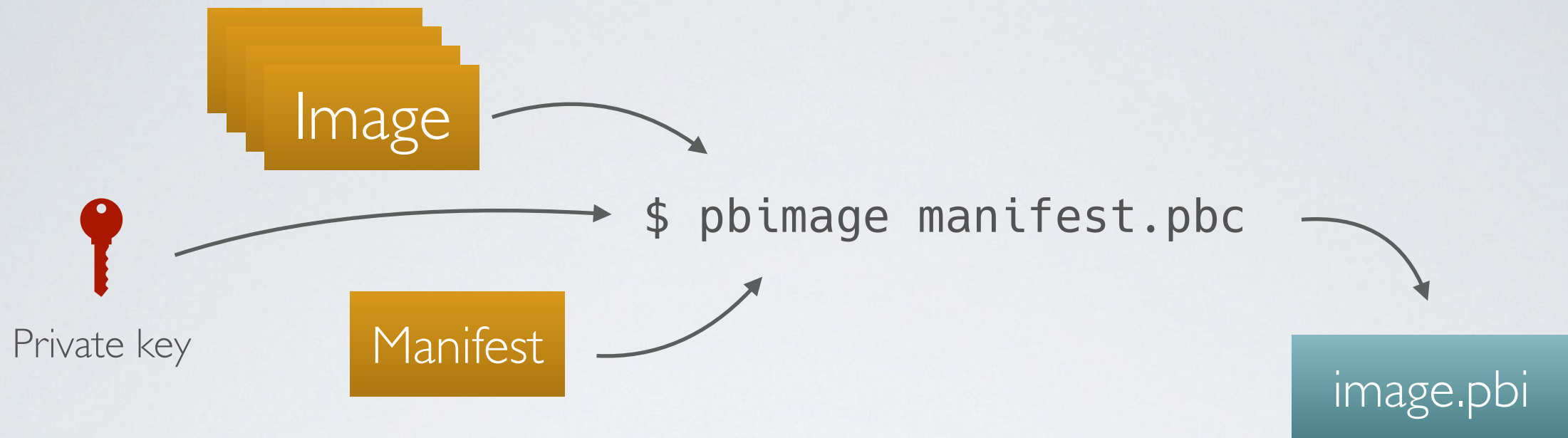


```
struct pb_image_hdr {  
    uint32_t header_magic;  
    uint32_t header_version;  
    uint32_t no_of_components;  
    uint32_t key_index;  
    uint32_t _reserved[23];  
    uint8_t sign[1024];  
    uint32_t sign_length;  
    uint32_t _reserved2[4];  
} __attribute__((packed));
```

```
struct pb_component_hdr {  
    uint32_t comp_header_version;  
    uint32_t component_type;  
    uint32_t load_addr_low;  
    uint32_t load_addr_high;  
    uint32_t component_size;  
    uint32_t component_offset;  
    uint32_t _reserved[16];  
} __attribute__((packed));
```

Max 16 components,
Image always contains 16
Component headers even if
All are not used

PBIMAGE TOOL



PB Image manifest

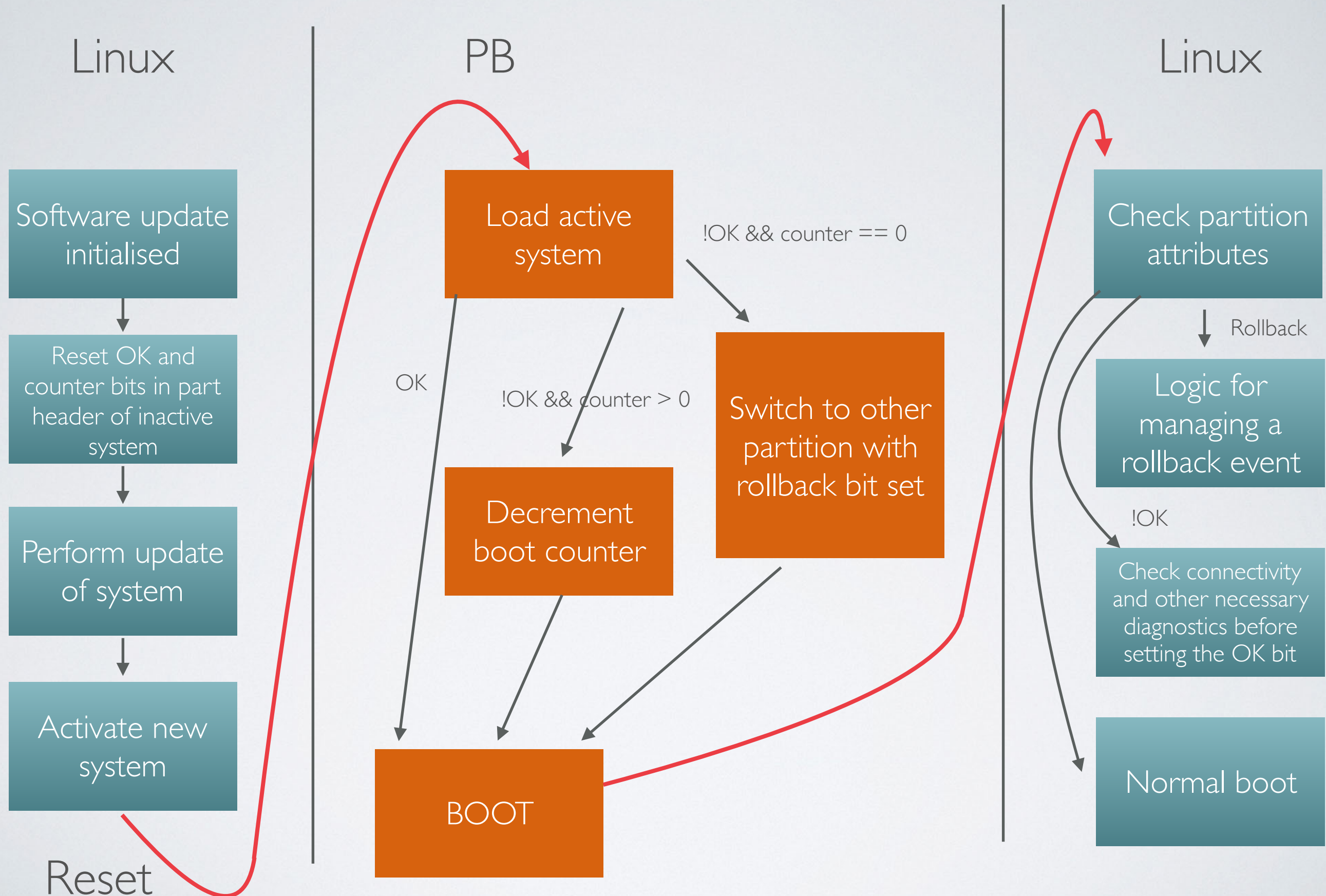
```
[pbimage]
key_index = 1
key_source = ../pki/prod_rsa_private.der
output = jiffy.pbi
```

```
[component]
type = ATF
load_addr = 0x80000000
file = /work/imx-atf/build/imx8qxp/release/bl31.bin
```

```
[component]
type = DT
load_addr = 0x82000000
file = /work/linux-imx/arch/arm64/boot/dts/freescale/jiffy.dtb
```

```
[component]
type = LINUX
load_addr = 0x82020000
file = /work/linux-imx/arch/arm64/boot/Image
```

A/B system update and rollback logic



Supported boot modes for IMX8X / M

- ATF+DT+LINUX
- ATF+DT+LINUX+TEE

v0.3

- Loading auxiliary M4 core

15 MByte boot image on IMX8X

- Power on reset
- Bootloader init
- Blockdev read
- SHA256
- RSA Signature

