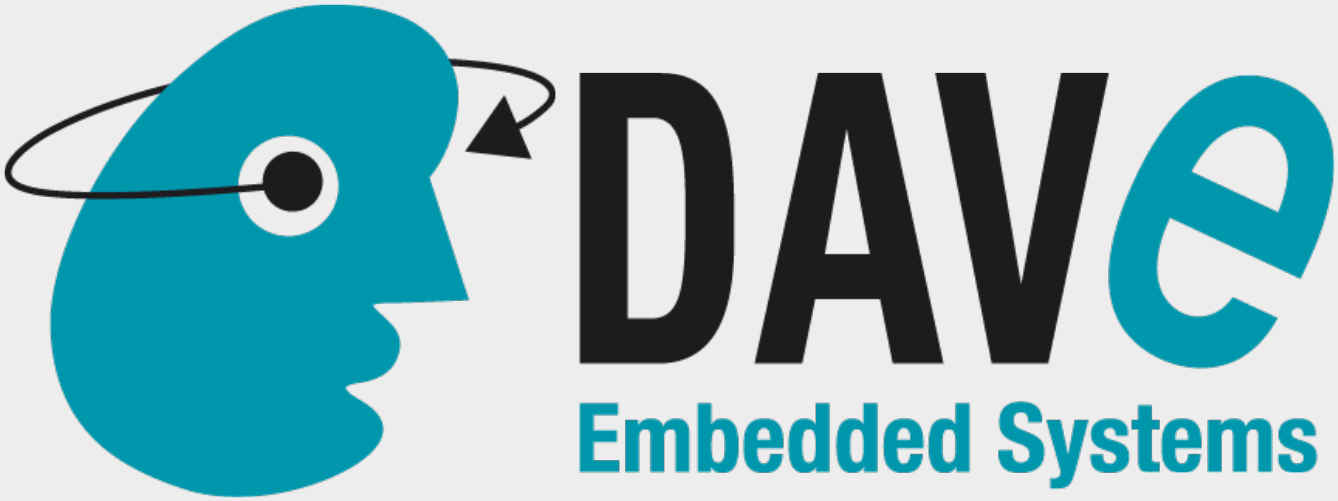
DAVE Embedded Systems



The AOSP and AAOS ~~May~~ November meetup

Securing iMX6 Android Devices

An introduction to integrating iMX High Assurance Boot into AOSP chain of trust to secure real world products

Agenda

- whoami
- what is (iMX6) Secure Boot and why use it
- AOSP chain of trust overview
  - build/validating AOSP images with iMX HAB
- put it all together to make a product with all this stuff

Too much to fit 20 minutes?! :-(

DAVE Embedded Systems

whoami

- Software development manager @ DAVE Embedded System
- ~19y in Embedded Linux Systems SW
  - hardware support
  - Yocto, AOSP, RealTime
- experience in different IT fields (sysadmin, IT automation, DBMS/ERP, cybersecurity background (CEH))

DAVE Embedded Systems

Disclaimer

- I’m NOT a (cyber) security expert
- something might be a bit outdated (M6 on iMX6), but general concept are valid

- review everything by a third party consultant
  - security audit is a must!

DAVE Embedded Systems

(iMX6) Secure Boot – why?

- what is a “secure” device?
- application (network?) security vs physical security
  - lots of embedded devices are physically accessible in a hostile environment
- what do you want to “protect”?
  - software IP?
  - hardware? (from running other software?)
  - “cloud” credential (keys?)
- from whom?
  - causal damage, script kiddies, cybercriminal, government?

how much can/must we invest in security?

DAVE Embedded Systems

E.g. protect “provision keys”

- store in SOC memory
- don’t let SOC execute untrusted (lowlevel) software
- don’t let (highlevel, untrusted) software access lowlevel resources

DAVE Embedded Systems

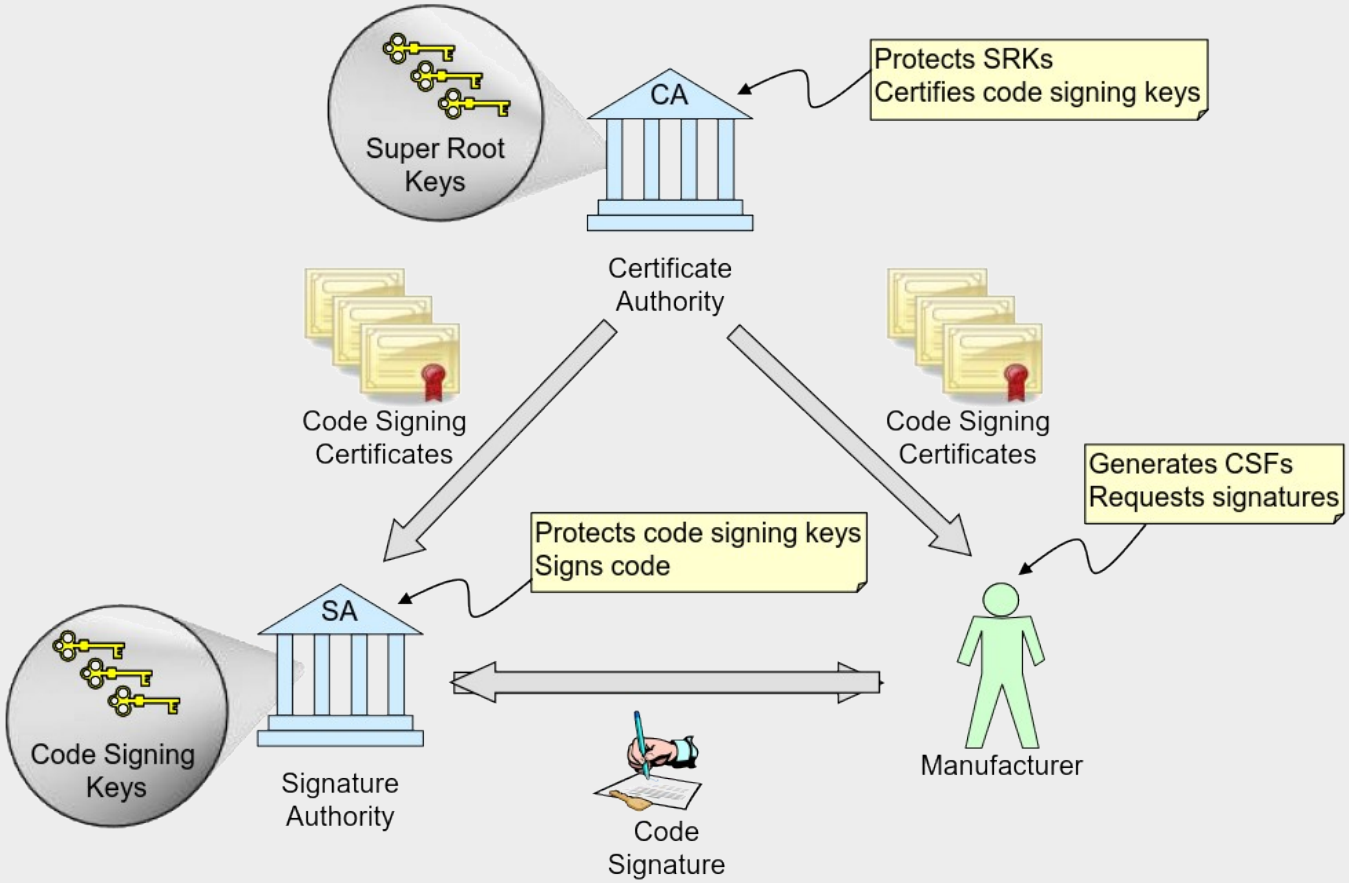
Secure Boot

- we need:
  - hardware support as root of trust
  - PKI: https://en.wikipedia.org/wiki/Public_key_infrastructure

DAVE Embedded Systems

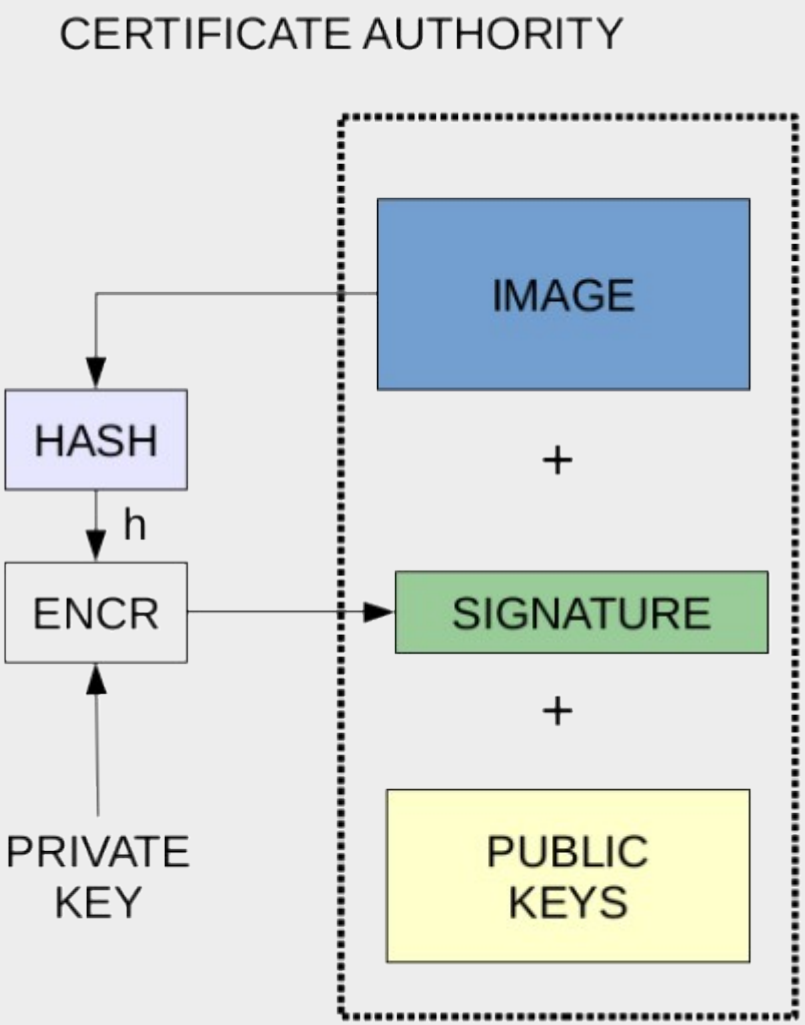
Introducing HAB

iMX → High Assurance Boot



DAVE Embedded Systems

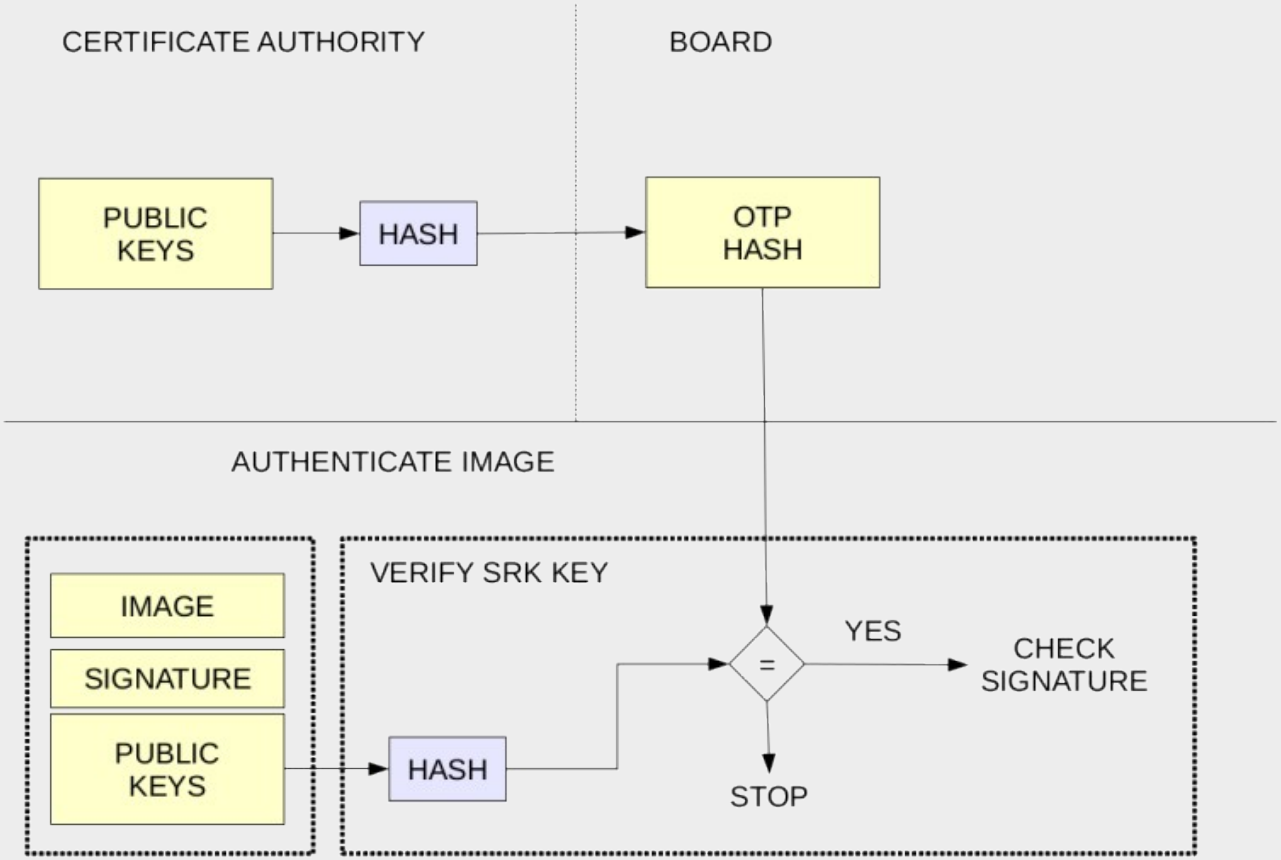
HAB: signing images



DAVE Embedded Systems

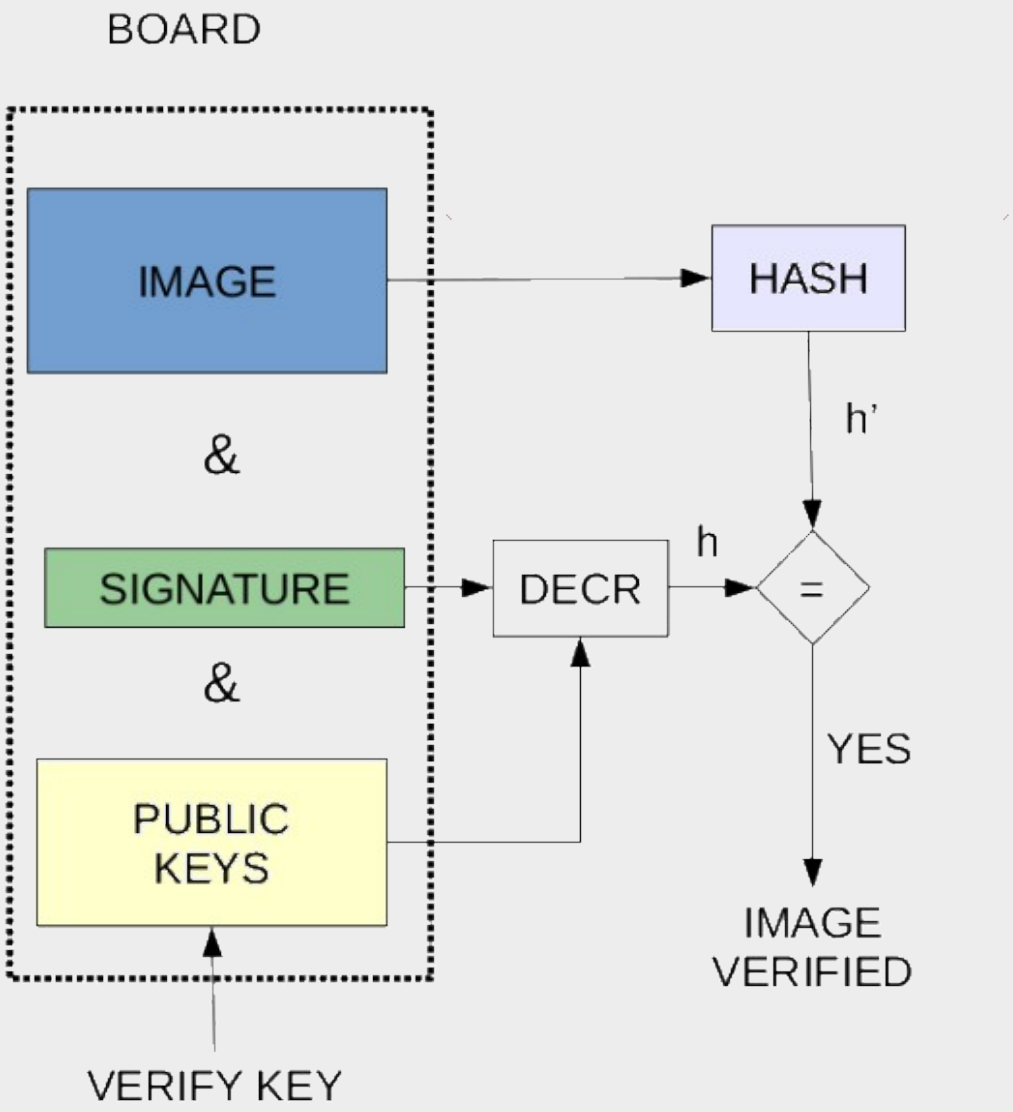
HAB: checking public key

- Public key are NOT stored into device



DAVE Embedded Systems

HAB: check signature



DAVE Embedded Systems

Nothing is perfect..

- As any software has bug, any hardware has errata..
- https://blog.quarkslab.com/vulnerabilities-in-high-assurance-boot-of-nxp-imx-microprocessors.html
- https://github.com/usbarmory/usbarmory/blob/master/software/secure_boot/Security_Advisory-Ref_QBVR2017-0001.txt
  - CVE-2017-7936 Buffer overflow in SDP recovery mode
  - CVE-2017-7932

DAVE Embedded Systems

HAB setup

- Get NXP Code Signing Tool (**CST**)
  - suggestion: track your copy with git
- Generate Certificate Authority (CA), Super Root Keys (SRK)…
  - 4 signing key are generated
  - can be revoked later
- Start signing images

PKI setup

```
dvdk@vagrant-ubuntu-trusty-64:~/lynx/hab/cst-2.3.3/keys$ cat serial
12345684
dvdk@vagrant-ubuntu-trusty-64:~/lynx/hab/cst-2.3.3/keys$ cat key_pass.txt
HAB Secure boot demo
HAB Secure boot demo
dvdk@vagrant-ubuntu-trusty-64:~/lynx/hab/cst-2.3.3/keys$
```

```
dvdk@vagrant-ubuntu-trusty-64:~/lynx/hab/cst-2.3.3/keys$ ./hab4_pki_tree.sh

    ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
    This script is a part of the Code signing tools for Freescale's
    High Assurance Boot.  It generates a basic PKI tree.  The PKI
    tree consists of one or more Super Root Keys (SRK), with each
    SRK having two subordinate keys:
[...]
```

DAVE Embedded Systems

HAB setup output

- “a bounch of .pem”
  - these are the one required to “sign” images (private!)
  - where to store? (git for development)
- SRK fuse values → public

```
dvdk@vagrant-ubuntu-trusty-64:~/lynx/hab/cst-2.3.3/crts$ hexdump -e '/4 "0x"' -e '/4 "%X""\
n"' SRK_1_2_3_4_fuse.bin
0x56D3468C
0x2313CE16
0xF374DA2D
0xB0843A55
0x74F39B9
0xEF12FBA6
0x2555E044
0x242C46B8
```

DAVE Embedded Systems

Flashing SRK fuse

```
i=0
for l in $(hexdump -e '/4 "0x"' -e '/4 "%X""\n"' SRK_1_2_3_4_fuse.bin)
do
     echo "echo $l > /sys/fsl_otp/HW_OCOTP_SRK$i"
     i=$(expr $i + 1)
done
```

```
echo 0x1E8BC131 > /sys/fsl_otp/HW_OCOTP_SRK0
echo 0x95CC7880 > /sys/fsl_otp/HW_OCOTP_SRK1
echo 0x1121334A > /sys/fsl_otp/HW_OCOTP_SRK2
echo 0x5906BB57 > /sys/fsl_otp/HW_OCOTP_SRK3
echo 0x46920123 > /sys/fsl_otp/HW_OCOTP_SRK4
echo 0x76D67934 > /sys/fsl_otp/HW_OCOTP_SRK5
echo 0xF98B606D > /sys/fsl_otp/HW_OCOTP_SRK6
echo 0x77615E98 > /sys/fsl_otp/HW_OCOTP_SRK7
```

LOCK after programming to avoid corruption

```
root@imx6qxelk:~# LOCK=$((1 << 14))
root@imx6qxelk:~# printf '0x%x\n' $LOCK
0x4000
root@imx6qxelk:~# echo $LOCK > /sys/fsl_otp/HW_OCOTP_LOCK
```

DAVE Embedded Systems

One-Time-Programmable (OTP) fusing tool

https://github.com/usbarmory/crucible

```
crucible -s -m IMX6UL -r 1 -l
...
 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00  OCOTP_CFG1
┌──────────────────────┬──────────────────────┬──────────────┬──────────────────────────────────┐                Bank:0 Word:2
│DIE-X-CORDINATE       │DIE-Y-CORDINATE       │WAFER_NO      │LOT_NO_ENC[42:32]                 │ R: 0x00000008
└──────────────────────┴──────────────────────┴──────────────┴──────────────────────────────────┘ W: 0x00000008
 31 ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ ------ 00  UNIQUE_ID[63:32]
 31 ------ ------ ------ ------ ------ ------ 24 ──────────────────────────────────────────────────────────────────────────────  DIE-X-CORDINATE
                       23 ------ ------ ------ ------ ------ ------ 16 ──────────────────────────────────────────────────────────  DIE-Y-CORDINATE
                                                 15 ------ ------ ------ 11 ──────────────────────────────────────────────────  WAFER_NO
                                                                          10 ------ ------ ------ ------ ------ ------ ------ ------ ------ 00  LOT_NO_ENC[42:32]
...
```

Secure Boot in action

- When OTP are programmed, HAB “start” working

Valid Image boot

```
=> hab_status
Secure boot disabled
HAB Configuration: 0xf0, HAB
State: 0x66
No HAB Events Found!
```

Invalid Image boot

```
=> hab_status
Secure boot disabled
HAB Configuration: 0xf0, HAB State: 0x66
--------- HAB Event 1 -----------------
event data:
        0xdb 0x00 0x14 0x42 0x33 0x28 0x33
0x00
        0x00 0x00 0x00 0x0f 0x87 0x7f 0xbb
0xd4
        0x00 0x09 0x00 0x00
STS = HAB_FAILURE (0x33)
RSN = HAB_INV_CALL (0x28)
CTX = HAB_CTX_TARGET (0x33)
ENG = HAB_ENG_ANY (0x00)
```

DAVE Embedded Systems

Last step – lock device

- After this fuse is written, BootROM will start only signed firmware
- If firmware is not (correctly) signed “nothing” happens
  - if you program the wrong SRKx now SOC is bricked
  - SRKx better to be locked!

```
echo 2 > /sys/fsl_otp/HW_OCOTP_CFG5
```

Do this only when you see “no HAB event found”!

DAVE Embedded Systems

AOSP chain of trust - 1

- until now we have a valid bootloader, what's left?
  - boot/recovery image
    - read-only big binaries
  - system partition
    - ...
  - data
    - ...

DAVE Embedded Systems

Check images – boota

- drivers/usb/gadget/f_fastboot.c:do_boota()

```
#ifdef CONFIG_SECURE_BOOT_OS_AUTHENTICATE
        extern uint32_t authenticate_image(uint32_t ddr_start,
                uint32_t image_size);

        if (authenticate_image(load_addr, image_size)) {
            printf("Authenticate OK\n");
        } else {
            printf("Authenticate image Fail, boot aborted\n\n");
            return 1;
        }
#endif /* CONFIG_SECURE_BOOT_OS_AUTHENTICATE*/
```

DAVE Embedded Systems

Check image → call HW

- arch/arm/imx-common/hab.c:authenticate_image()

```
uint32_t authenticate_image(uint32_t ddr_start, uint32_t image_size)
{

[...]
     puts("\nCalling authenticate_image in ROM\n");
     printf("\tivt_offset = 0x%x\n", ivt_offset);
     printf("\tstart = 0x%08lx\n", start);
     printf("\tbytes = 0x%x\n", bytes);

[...]

     load_addr = (uint32_t)hab_rvt_authenticate_image(
          HAB_CID_UBOOT,
          ivt_offset, (void **)&start,
          (size_t *)&bytes, NULL);

     if (hab_rvt_exit() != HAB_SUCCESS) {
          puts("hab exit function fail\n");
          load_addr = 0;
     }
}
```

DAVE Embedded Systems

Sign AOSP images

- into build/core/Makefile
- sign image and overwrite the unsigned

```
diff --git a/core/Makefile b/core/Makefile
index de28d568f..83e9d4eb4 100644
--- a/core/Makefile
+++ b/core/Makefile
@@ -515,6 +515,8 @@ $(INSTALLED_BOOTIMAGE_TARGET): $(MKBOOTIMG) $(INTERNAL_BOOTIMAGE_FILES)
$(BOOT_S
         $(BOOT_SIGNER) /boot $@ $(PRODUCTS.$
(INTERNAL_PRODUCT).PRODUCT_VERITY_SIGNING_KEY).pk8 $(PRODUCTS.$
(INTERNAL_PRODUCT).PRODUCT_VERITY_SIGNING_KEY).x509.pem $@; \
                 $(call assert-max-image-size,$@,$(BOARD_BOOTIMAGE_PARTITION_SIZE)); \
                 cp -f $@ $$BOOT_IMAGE_BOARD; \
+                LOADADDR=0x12000000 /opt/cst/linux64/sign-image.sh `pwd`/$
$BOOT_IMAGE_BOARD; \
+                cp -fv $${BOOT_IMAGE_BOARD}-ivt-signed $$BOOT_IMAGE_BOARD; \
         done

 .PHONY: bootimage-nodeps
@@ -941,6 +943,8 @@ define build-recoveryimage-target
     fi;\
     $(call assert-max-image-size,$(1),$(BOARD_RECOVERYIMAGE_PARTITION_SIZE)); \
     cp -f $(1) $$RECOVERY_IMAGE_BOARD; \
+    LOADADDR=0x12000000 /opt/cst/linux64/sign-image.sh `pwd`/$$RECOVERY_IMAGE_BOARD; \
+    cp -fv $${RECOVERY_IMAGE_BOARD}-ivt-signed $$RECOVERY_IMAGE_BOARD; \
   done
   @echo ----- Made recovery image: $(1) --------
 endef
```

DAVE Embedded Systems

AOSP chain of trust - 1

- until now we have a valid bootloader, what's left?
  - boot/recovery image
    - read-only big binaries
  - system partition
    - "read-only"
    - dmverity → signature in boot.img
  - data
    - ...

DAVE Embedded Systems

AOSP chain of trust - 3

- until now we have a valid bootloader, what's left?
  - boot/recovery image
    - read-only big binaries
  - system partition
    - "read-only"
    - dmverity → signature in boot.img
  - data
    - read-write
    - cannot be protected in this way → encryption
    - store encryption key (or part of..) in SoC OTP

DAVE Embedded Systems

Making a product

- Protecting certificates vs allow developing
  - keep production/dev separate
  - production build managed by product seller
  - automate!
- Production site security
  - SRK & (signed) images are “public”
  - API key (and similar) are “private”
    - generate and store directly on device

DAVE Embedded Systems

Making a product - repair

- Repair / rework
  - what if update brick a device?
  - what if a device must be repaired?
    - device testing need low level access
- need a safe way to “unlock” the device

Unlock the device

- Must be non persistent (we want to restore the device)
  - on phone unlock is persistent (void warranty)

Use a “unlocker” hw device

- device send a challenge to the *unlocker*
- the *unlocker* send the challenge to “a cloud” - here device unlock validation occours
- *unlocker* send the response to device
- device store the “unlocked” flag in a volatile memory area
  - allow booting untrusted images
  - allow root access

DAVE Embedded Systems

Another solution

- Throw it away!
  - if not too much expensive (e.g. SOM vs complex carrier board)

Close all doors!

- Serial console
  - TX is useful but may lack information
  - Try to find a way to enable console safely
- U-Boot environment
  - hard to protect → remove it!
  - use additional storage (persistent registers)
- JTAG
  - useful for HW inspection for repair
  - usually can be enabled with password

DAVE Embedded Systems

Close all doors - cont

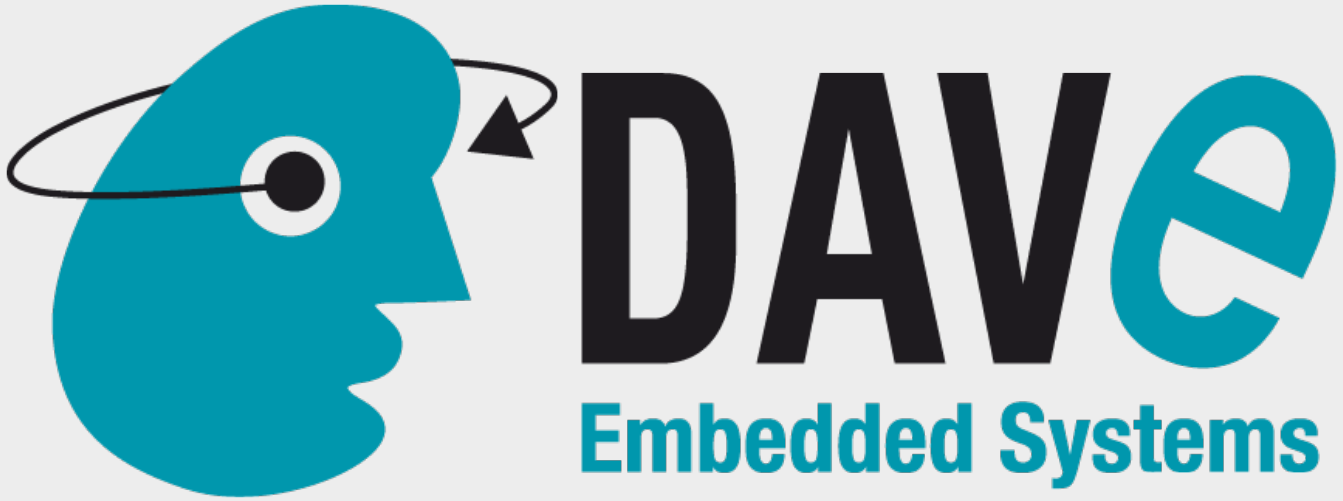
- high level software to remove/protect:
  - Settings, ADB
  - reduce attack surface, reduce unwanted damage (IEC 62443 – lev 1&2)

DAVE Embedded Systems

References

- IEC 62443
- https://wiki.dave.eu/index.php/XUELK-WP-001:_Secure_boot_on_iMX6UL
- iMX6 Security Reference Manual (NDA)
- https://source.android.com/security
  - https://source.android.com/security/verifiedboot

DAVE Embedded Systems



DAVE S.r.l.

Via Talponedo, 29/A
I-33080, Porcia (PN) Italy

Tel +39 0434 921215
Fax +39 0434 1994030

www.dave.eu
info@dave.eu
wiki.dave.eu