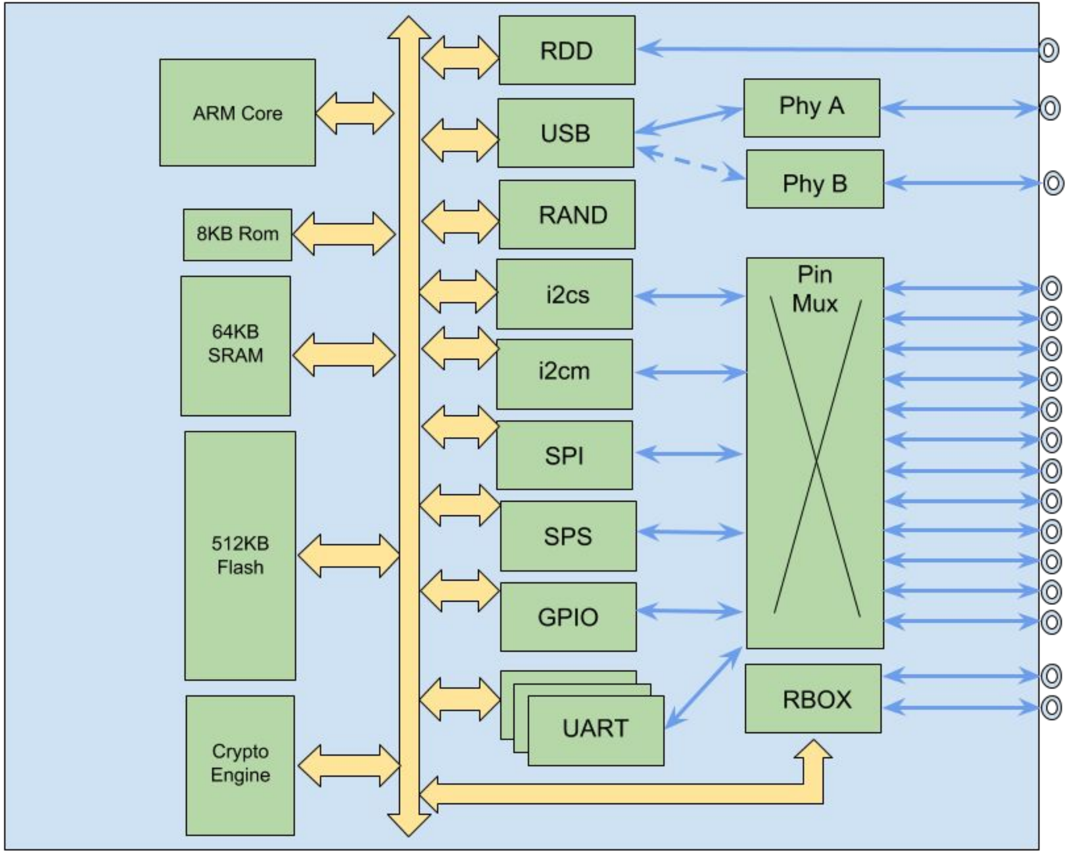
Google Security Chip H1
A member of the Titan family

Chrome OS Use Case

vbendeb@google.com

Block diagram

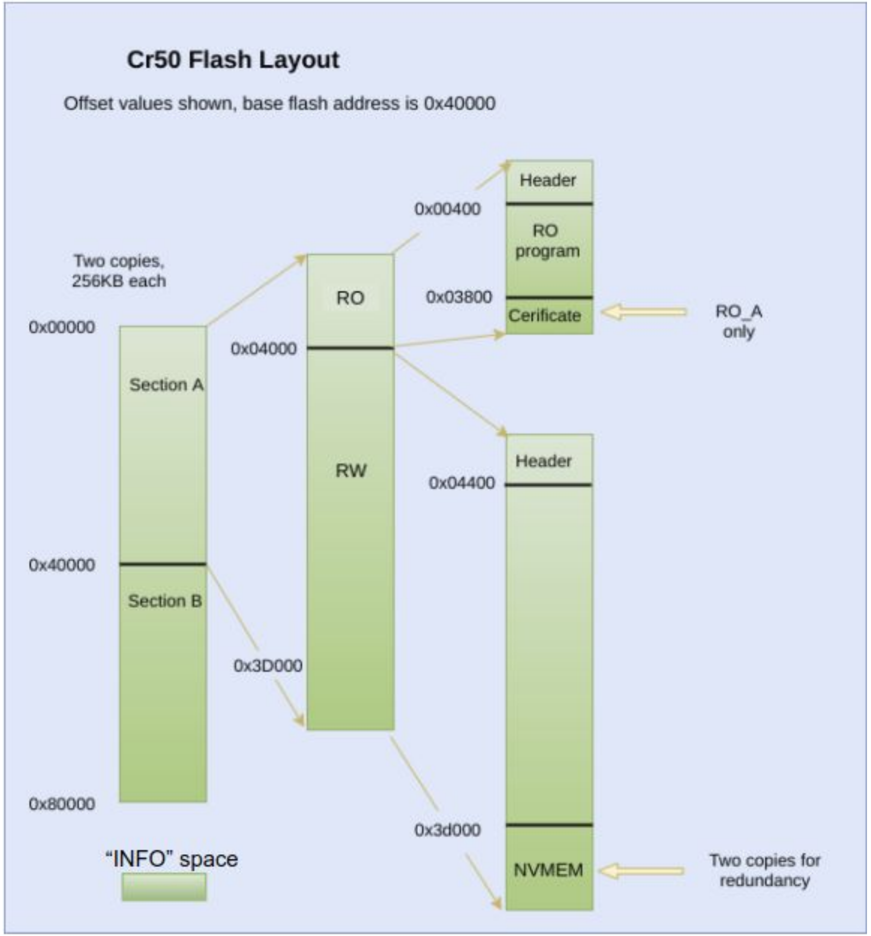
- ARM SC300 core
- 8kB boot ROM, 64kB SRAM, 512kB Flash
- USB 1.1 slave controller (USB2.0 FS)
- I2C master and slave controllers
- SPI master and slave controllers
- 3 UART channels
- 32 GPIO ports, 28 muxed pins
- 2 Timers
- Reset and power control (RBOX)
- Crypto Engine
- HW Random Number Generator
- RD Detection



Flash Memory Layout

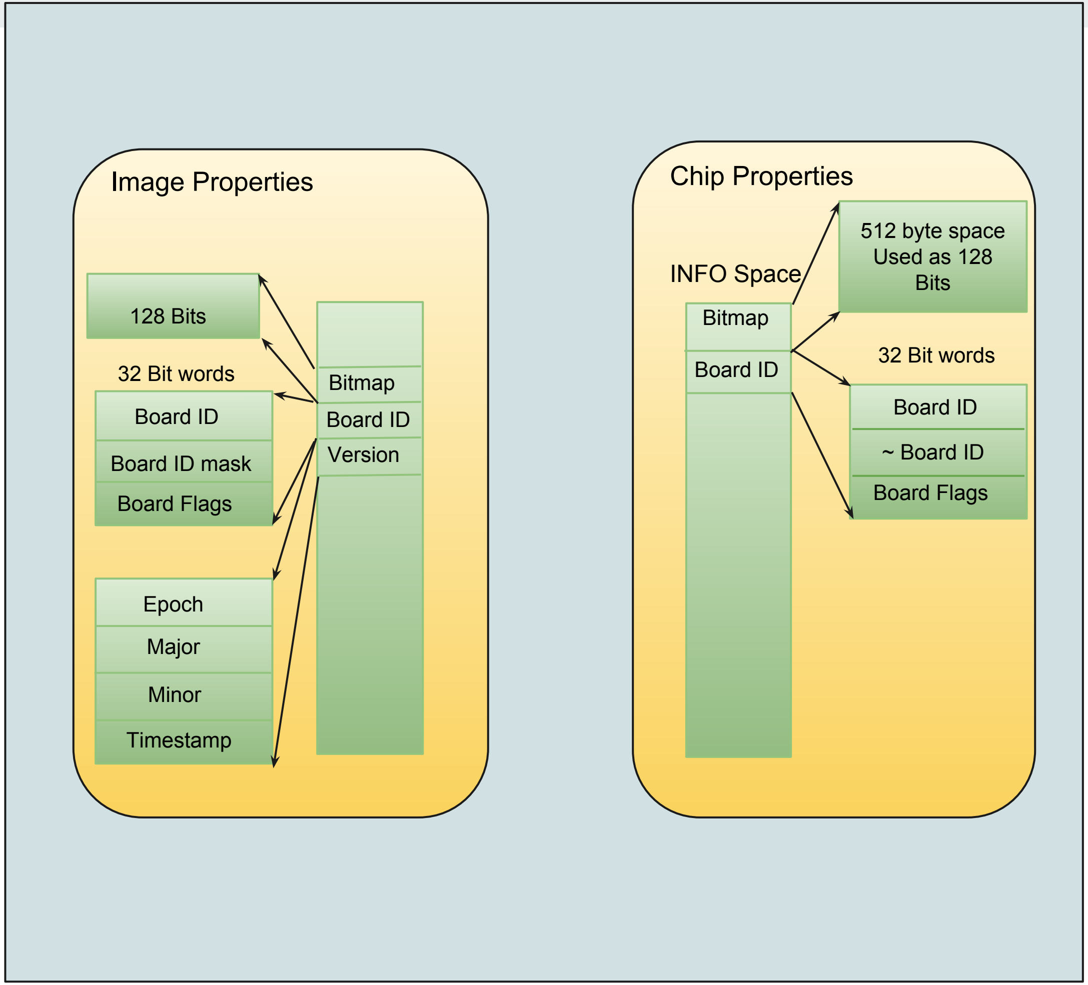
- Bootrom not shown
- Flash space split in two halves for redundancy
- Restricted access INFO space
- Header fields control boot flow
- Code is in Chrome OS EC repo*,
  - board files in `board/cr50`
  - chip files in `chip/g`

*https://chromium.googlesource.com/chromiumos/platform/ec



FW Updates

- Updates over USB or TPM
- Rollback protections
    - Header versioning scheme
    - Flash map bitmap
- Board ID and flags
- RO public key in ROM
- RW public key in RO
- Both ROM and RO allow for node locked signatures



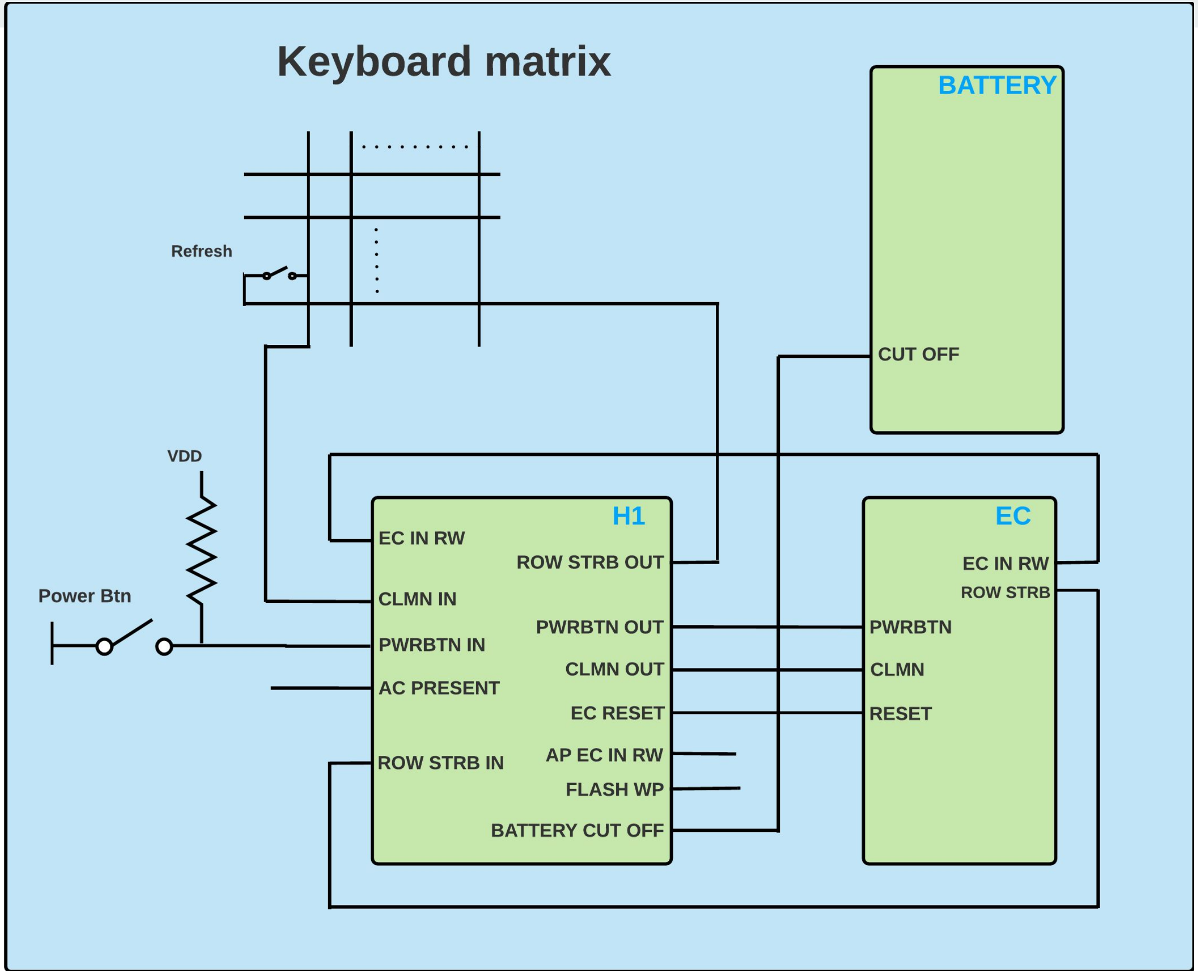
Major Functions

- Guaranteed Reset
- Battery cutoff
- Closed Case Debugging *
- Verified Boot (TPM Services)
- Support of various security features

* https://www.chromium.org/chromium-os/ccd

Reset and power

- Guaranteed EC reset and battery cutoff
- EC in RW latch (guaranteed recovery)
- SPI Flash write protection



TPM Interface to AP

- I2C or SPI
- Bootstrap options
- TPM Reset is not H1 reset

TPM Support Of Verified Boot

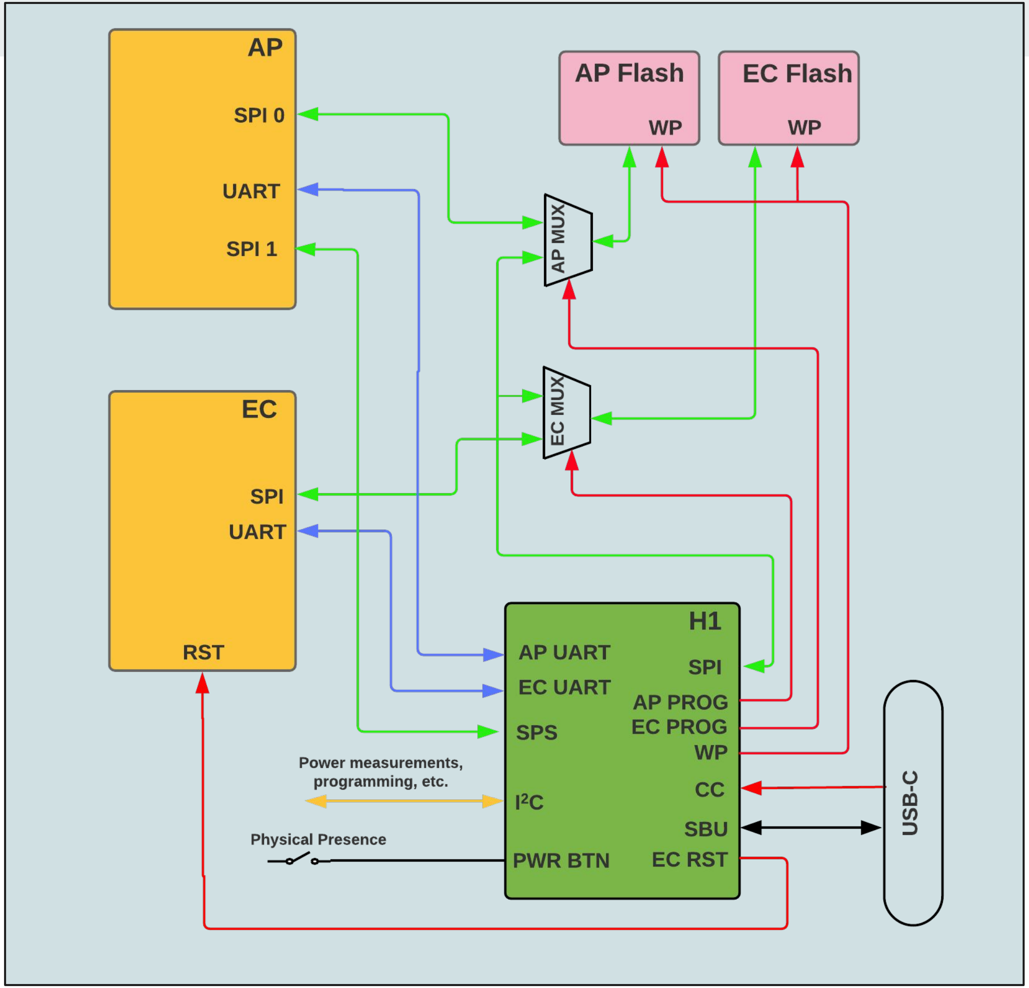
- Rollback counters for RW Firmware and Kernel
- MRC (Memory Reference Code) cache SHA
- FWMP (Firmware Management Parameters)
- Dev mode state

Closed Case Debugging

(Must be securely enabled with verified user physical presence)

- USB-C interface
- Triggered by SuzyQable*
- USB endpoints UART consoles
- CCD Capabilities
- Flash programming
- I2C debug and measurements
- Power button used for PP

*https://www.sparkfun.com/products/14746

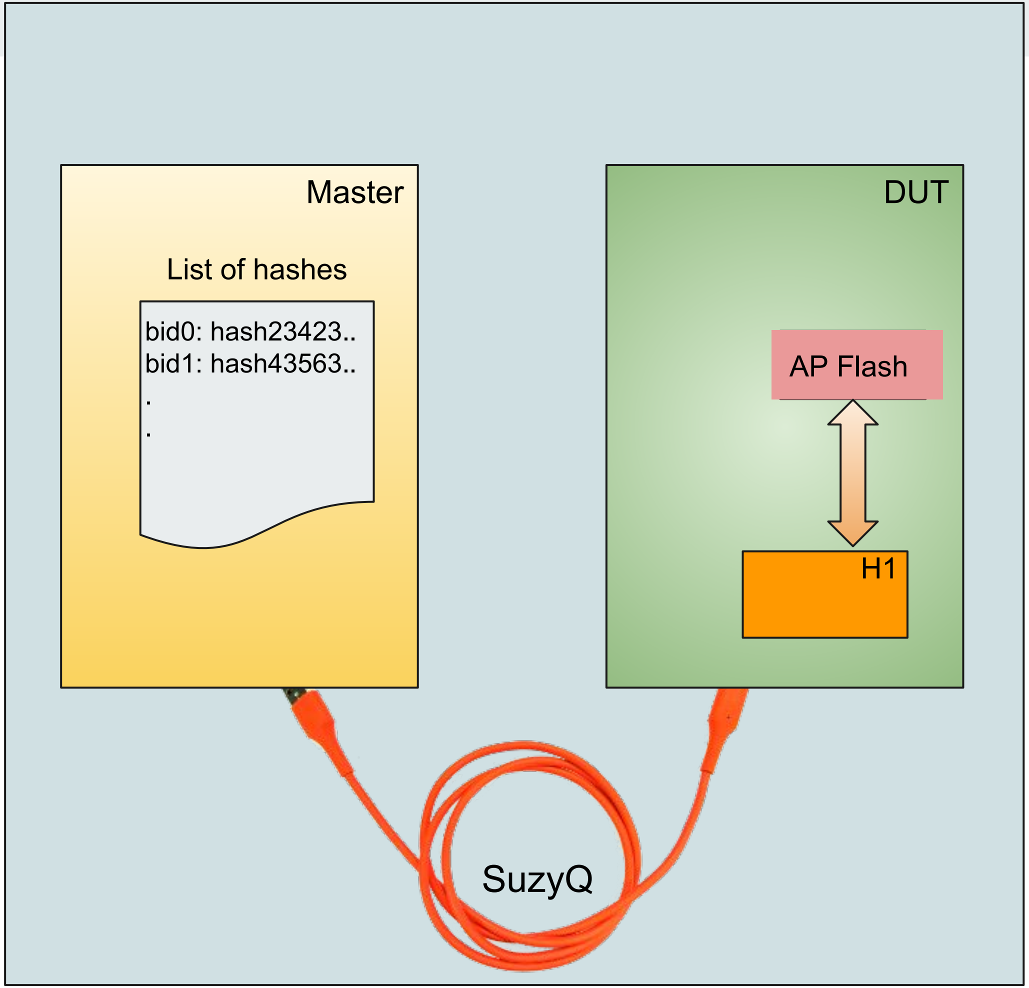


Security Features

- RMA Verification
- RMA Unlock
- Pin Weaver
- U2F Security Key

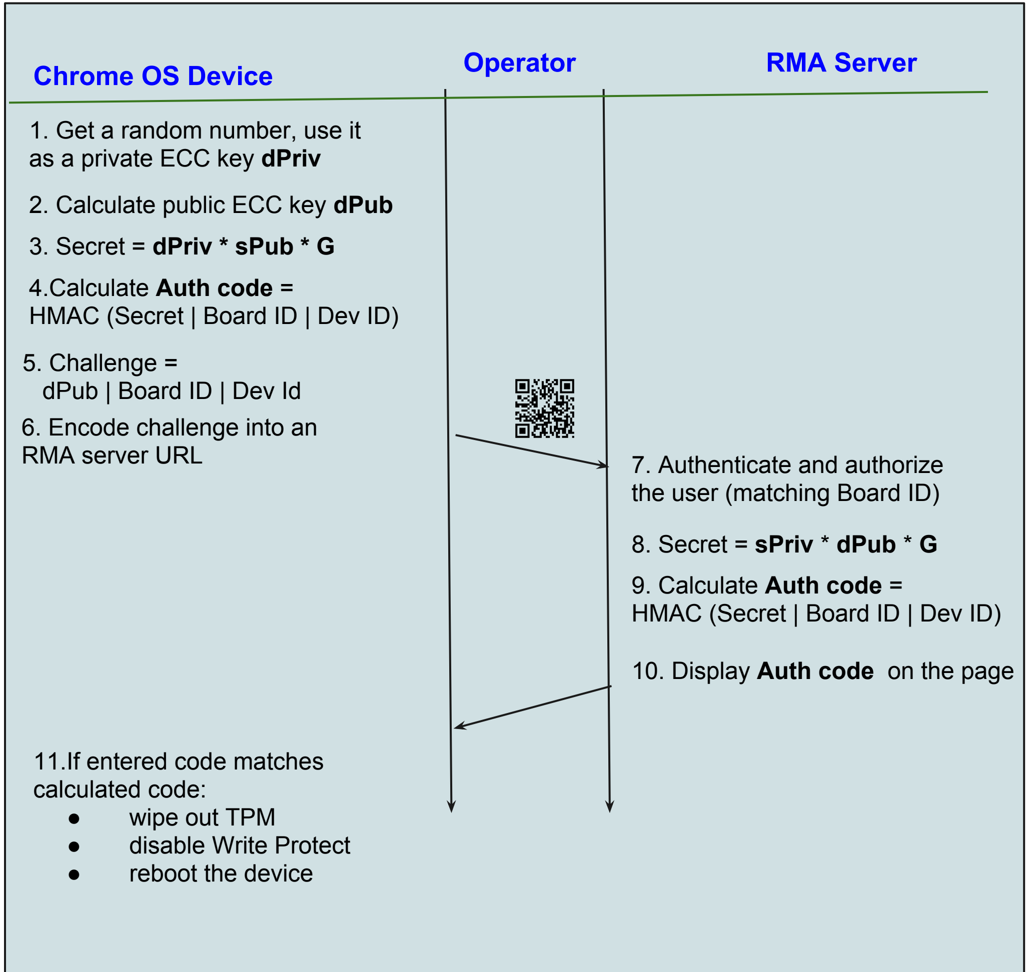
RMA Verification

- A Chrome OS device used as a master
- SuzyQuable connection to slave
- Update slave if necessary
- Verification of AP and EC firmware
- Hashes keyed by Board ID



RMA Unlock

- Uses ECC Diffie-Hellman
- Server account requires U2F
- Facilitates device servicing by disabling WP



Chrome OS Device

Operator

RMA Server

1. Get a random number, use it as a private ECC key **dPriv**

2. Calculate public ECC key **dPub**

3. Secret = **dPriv * sPub * G**

4.Calculate **Auth code** = HMAC (Secret | Board ID | Dev ID)

5. Challenge =
 dPub | Board ID | Dev Id

6. Encode challenge into an RMA server URL

7. Authenticate and authorize the user (matching Board ID)

8. Secret = **sPriv** * **dPub** * **G**

9. Calculate **Auth code** = HMAC (Secret | Board ID | Dev ID)

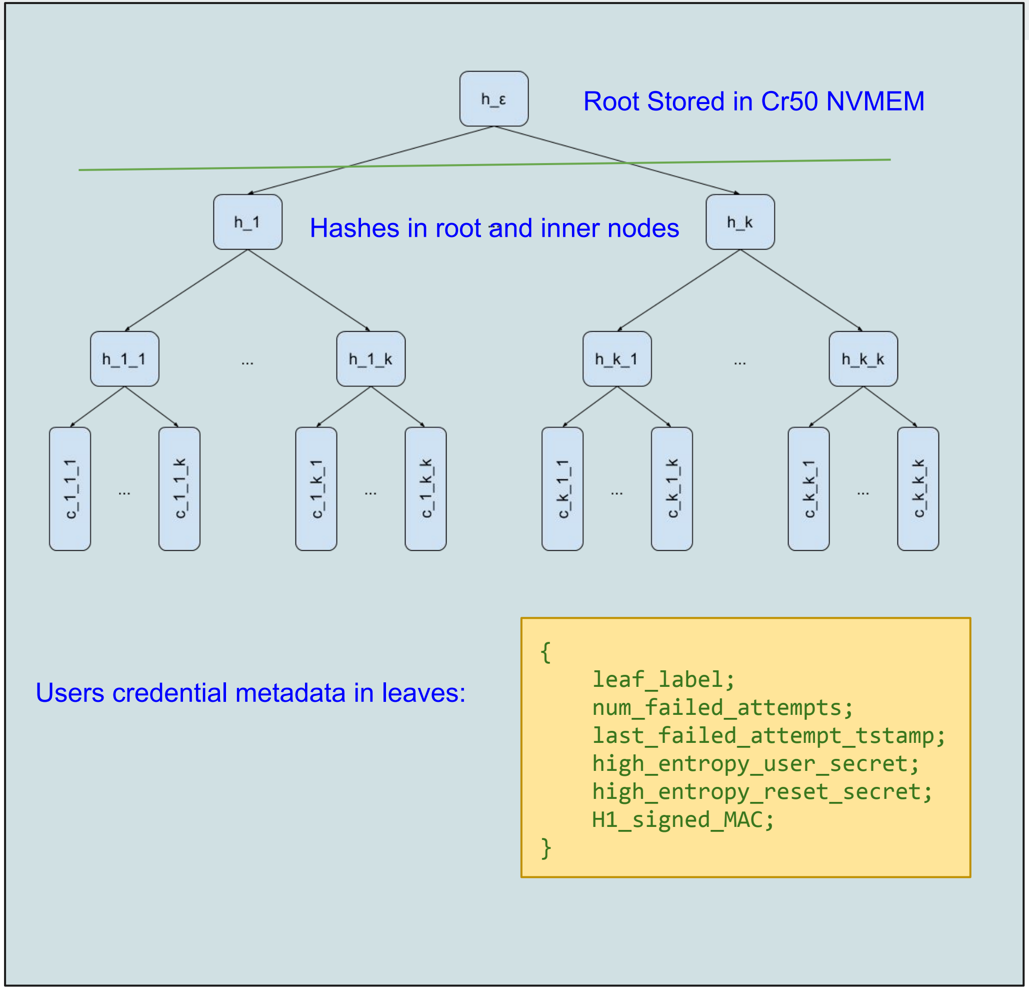
10. Display **Auth code** on the page

11.If entered code matches calculated code:

- wipe out TPM
- disable Write Protect
- reboot the device

Pin Login

- Low entropy password
- Multiple user accounts
- Both retry and rate limiting
- Merkle tree of descriptors
- Root stored on H1



Root Stored in Cr50 NVMEM

Hashes in root and inner nodes

Users credential metadata in leaves:

```
{
    leaf_label;
    num_failed_attempts;
    last_failed_attempt_tstamp;
    high_entropy_user_secret;
    high_entropy_reset_secret;
    H1_signed_MAC;
}
```

U2F Security Key

- Built in U2F
- Power button used for PP
- PK stored in H1

Questions?

Thank you!