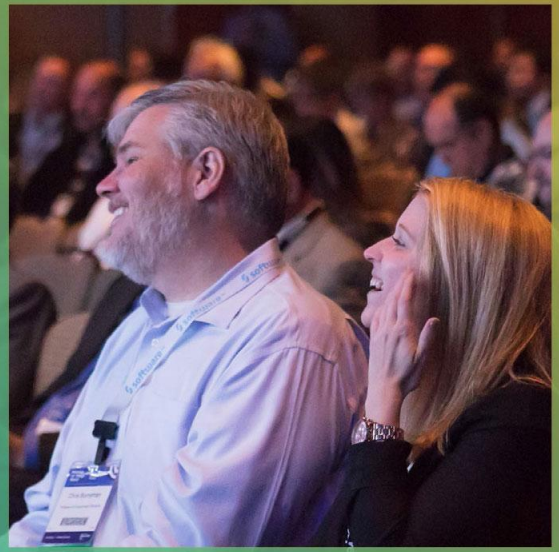
RISC-V Summit

December 3 - 6, 2018
Santa Clara Convention Center
CA, USA



REVOLUTIONIZING THE COMPUTING LANDSCAPE AND BEYOND.

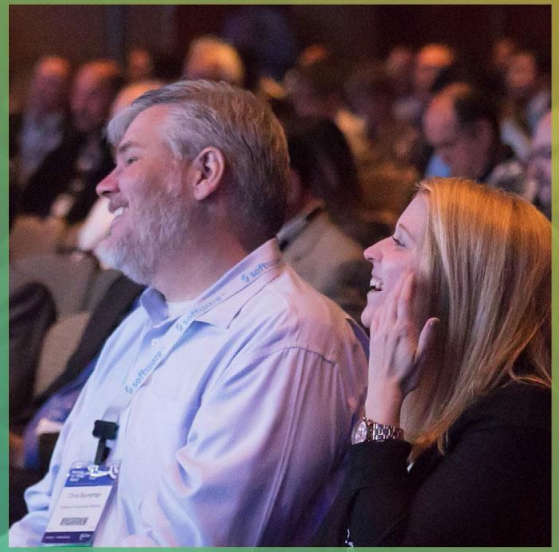
https://tmt.knect365.com/risc-v-summit

@risc_v

RISC-V Summit

SECURE BOOTSTRAPPING OF TRUSTED SOFTWARE IN RISC-V

Ilia Lebedev
Graduate Student
Massachusetts Institute of Technology



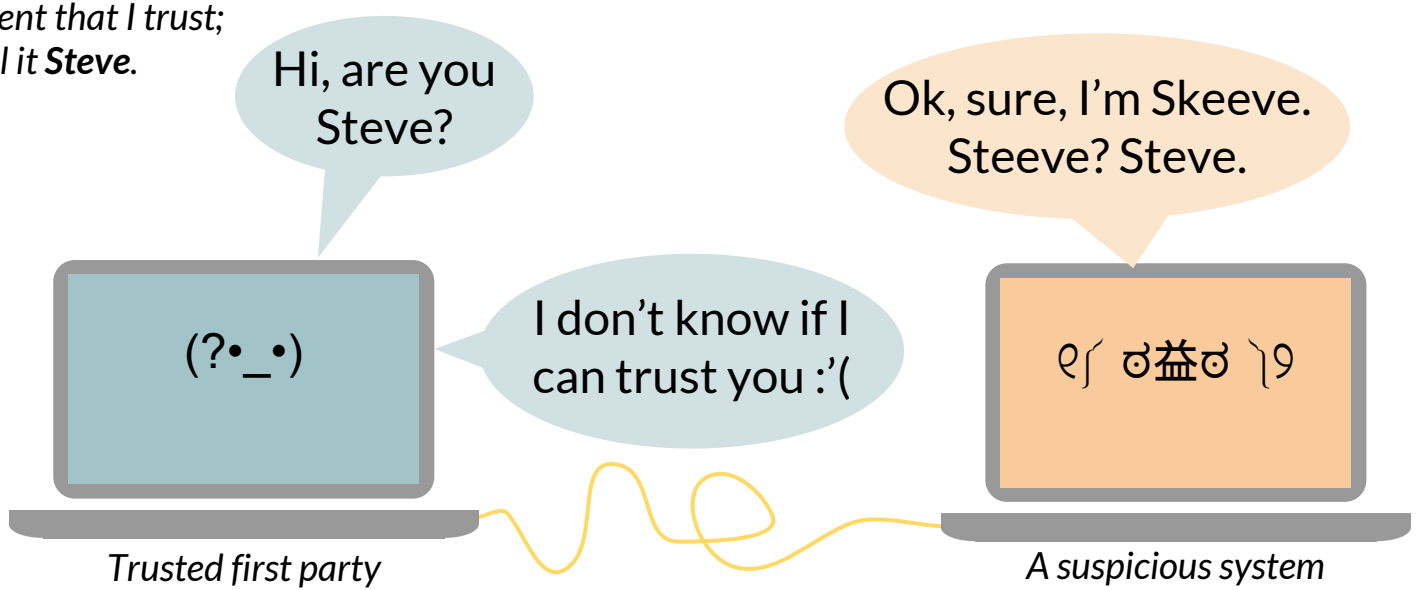
Secure Bootstrapping of Trusted Software in RISC-V

Ilia Lebedev, Kyle Hogan, Jules Drean, Srini Devadas - MIT

This work was partially funded by Delta Electronics, Analog Devices, DARPA & SPAWAR (N66001-15-C-4066), and DARPA SSITH (HR001118C0018).

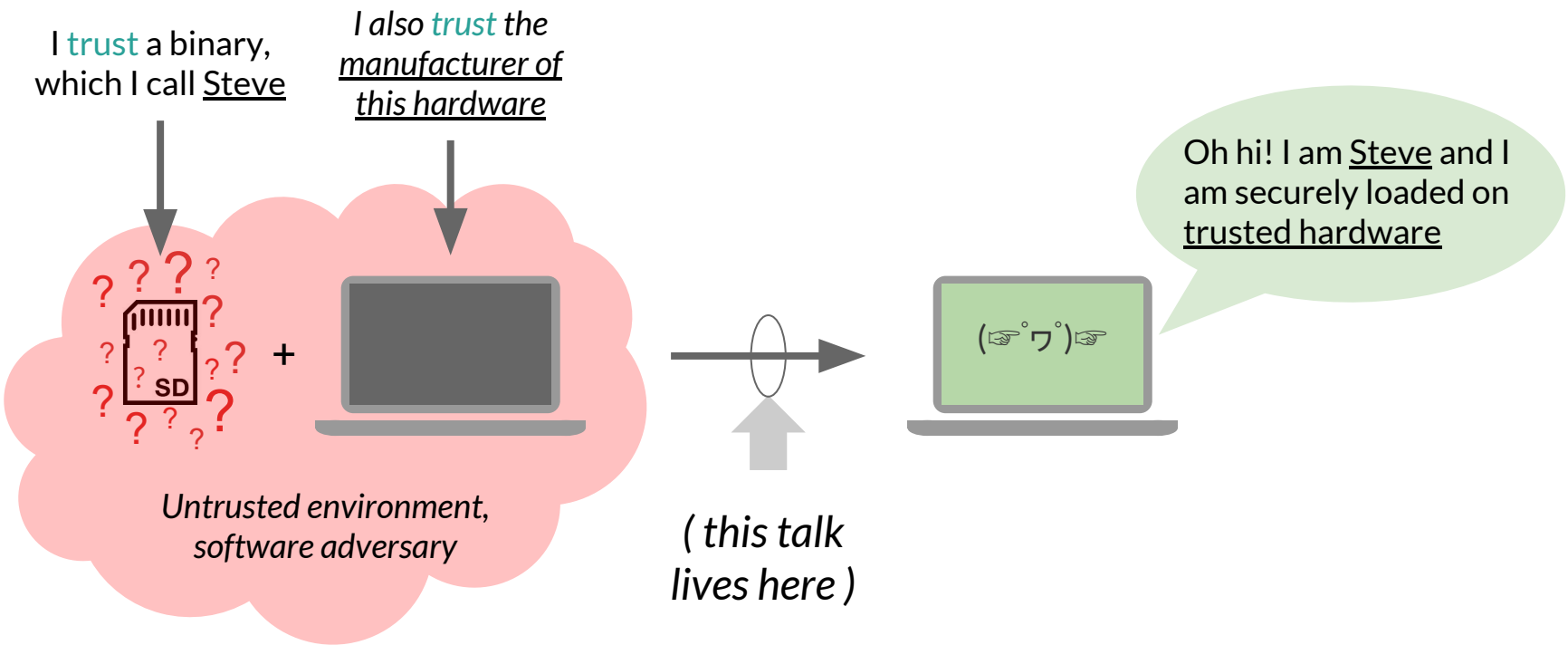
We often have no idea whether trust is justified

*I expect a specific software environment that I trust; I call it **Steve**.*

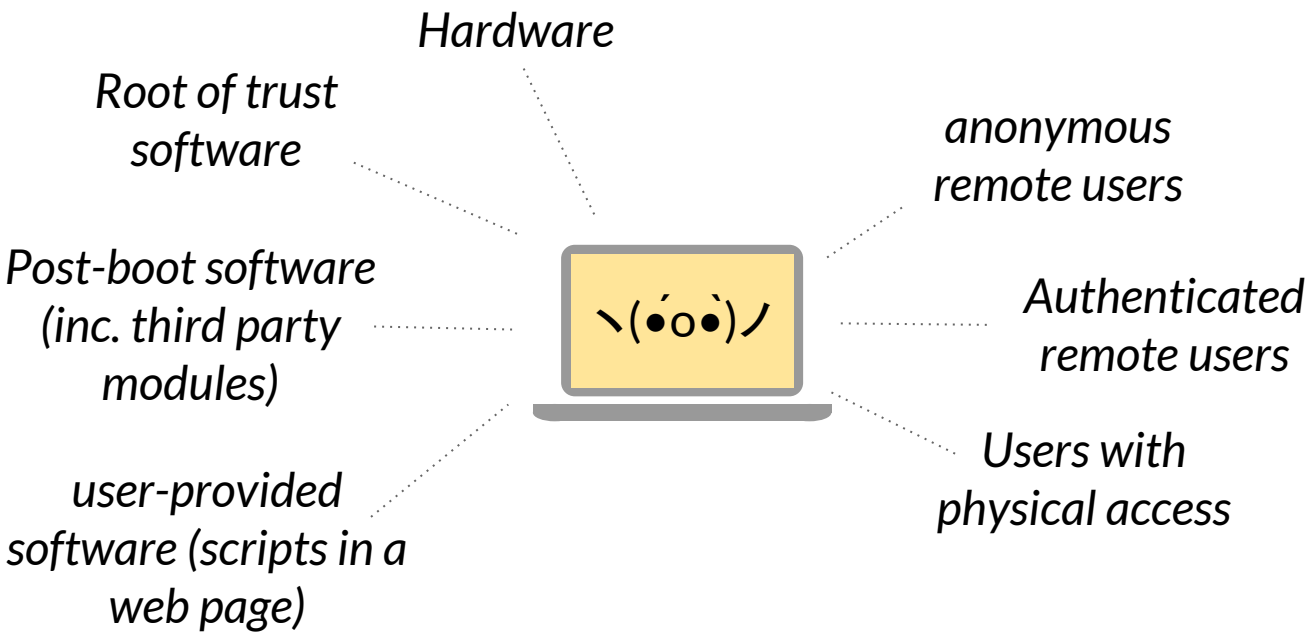


Secure Bootstrapping of Trusted Software in RISC-V

- https://medium.com/@ilia.lebedev/secure-boot-2d6e319b6978
- https://eprint.iacr.org/2018/427

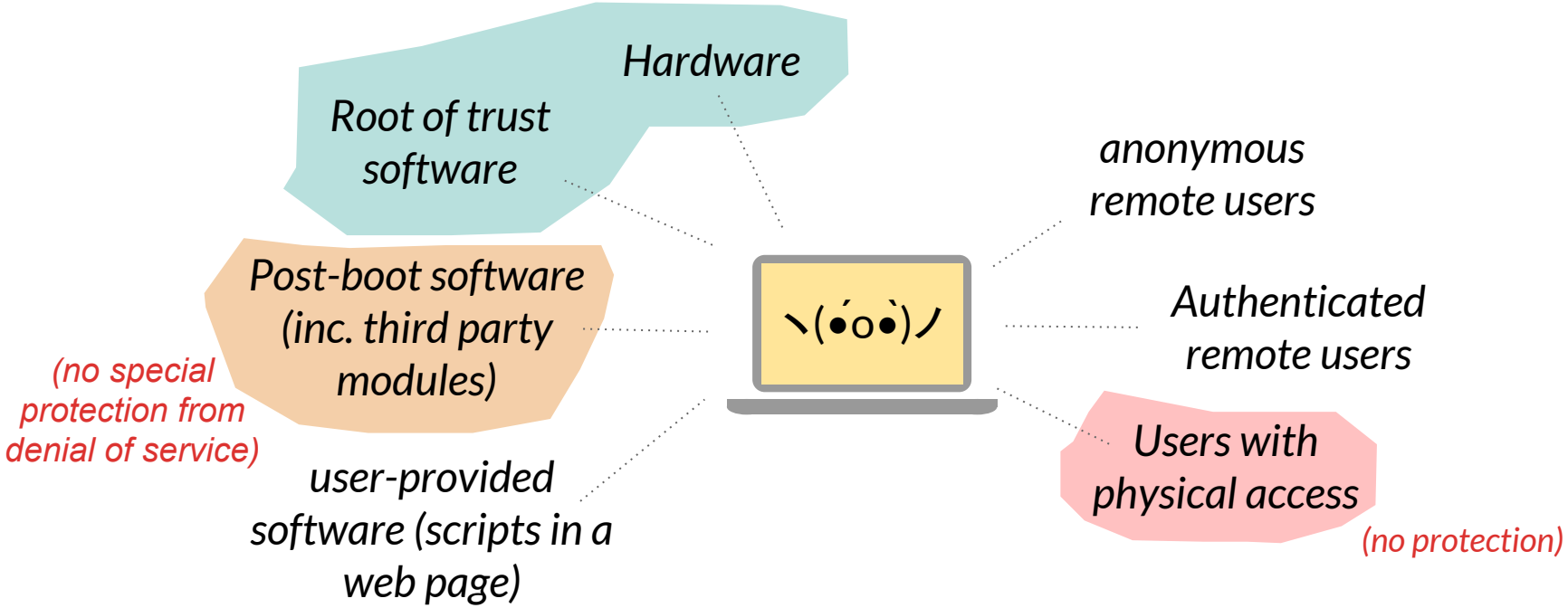


Consider trust: actors in a system (1/2)



How is each trusted?
What specific guarantees can be made?

Consider trust: actors in a system (2/2)



Guarantee integrity for loading a boot image in an untrusted setting, given pre-existing trust in HW, manufacturer, desired boot image

Threat Model

Unconditional trust:

- Manufacturer's keys, trusted HW
- Desired boot image (key privacy)
- … and you! (the first party)

Adversary: boot a different, malicious binary, interact with it

… but not tamper with the the hardware

… or extract its secret key

… or deny service

1: M is a trustworthy CA

We will document our chain of trust here:

Certificate Authority

Manufacturer

Manufacturer securely generates and stores SK_M

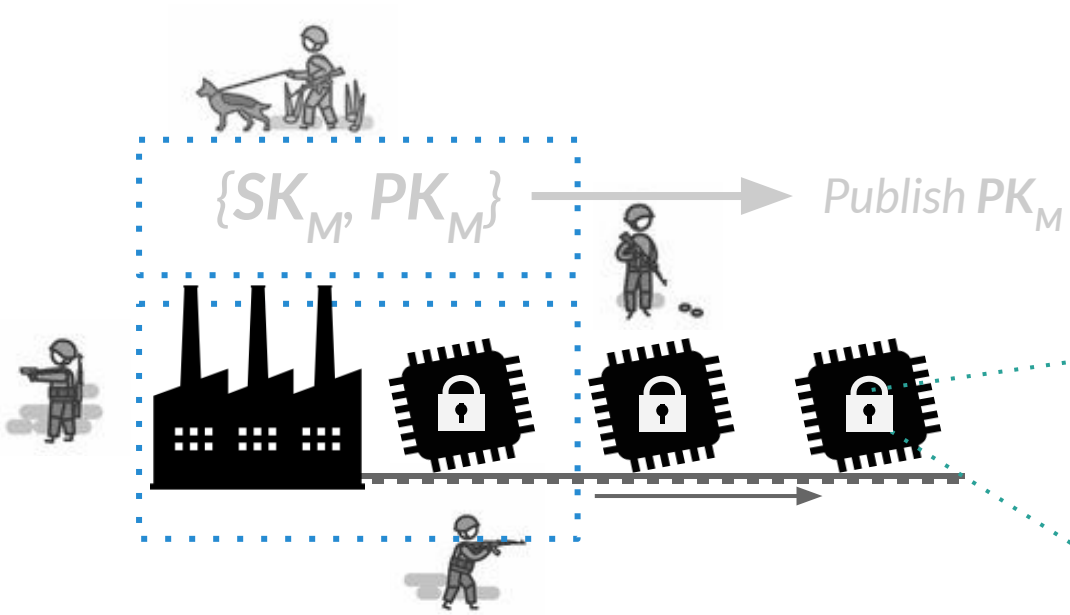
$\{SK_M, PK_M\}$

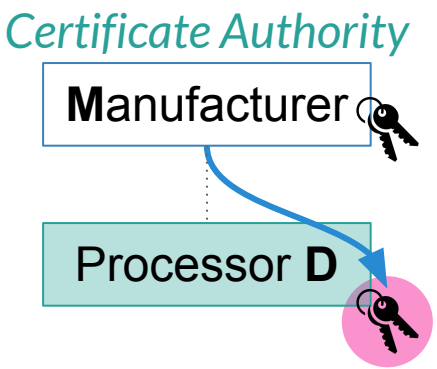
*ed25519 cryptosystem**

→ *Publish* PK_M

**RSA is also a reasonable choice*

2: M builds a trustworthy D with keys





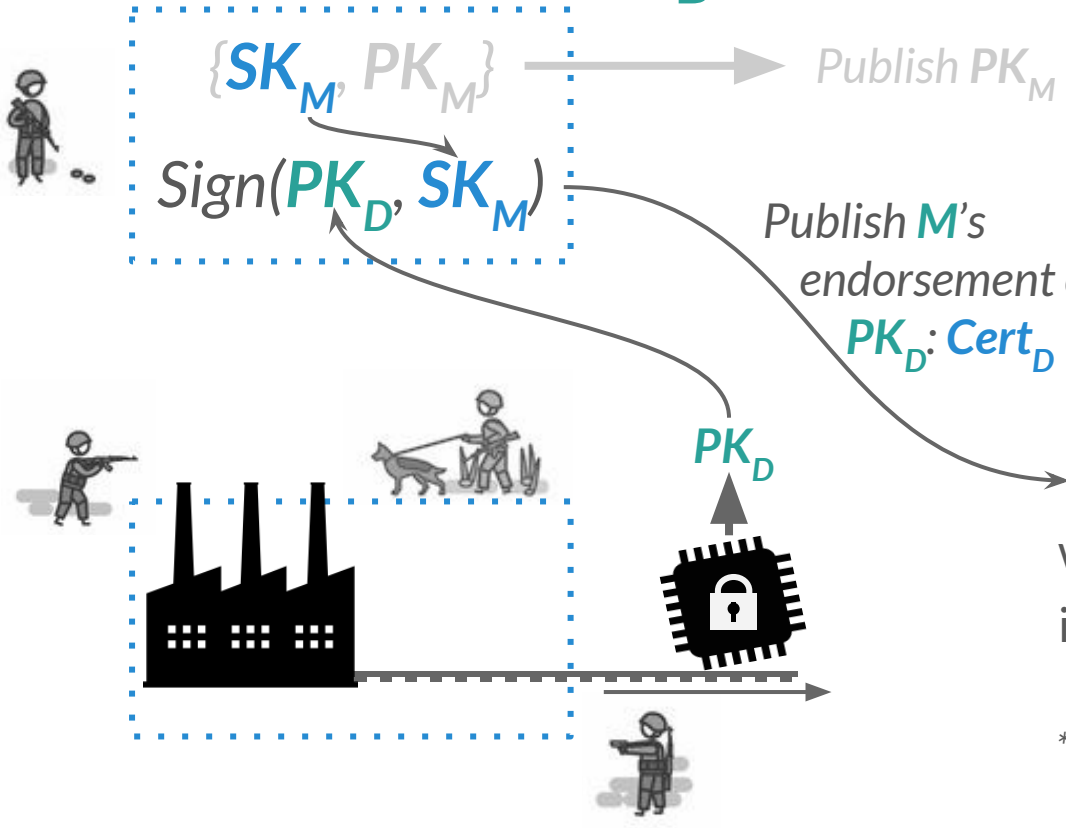
D has some properties:

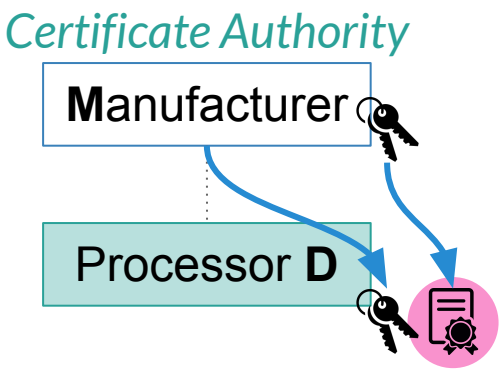
- Trustworthy **keys**
- No back doors
- A trusted **ROM**
- First instruction integrity

*D exposes SK_D via a special interface, and is hidden from normal software.

$\{SK_D^*, PK_D\}$

3: M certifies PK_D





Proves* that a device with access to SK_D is D manufactured by M.

*trust assumptions...

4: D executes from a root of trust at reset

D must guarantee first instruction integrity

1). **Erase** (or re-key) memory

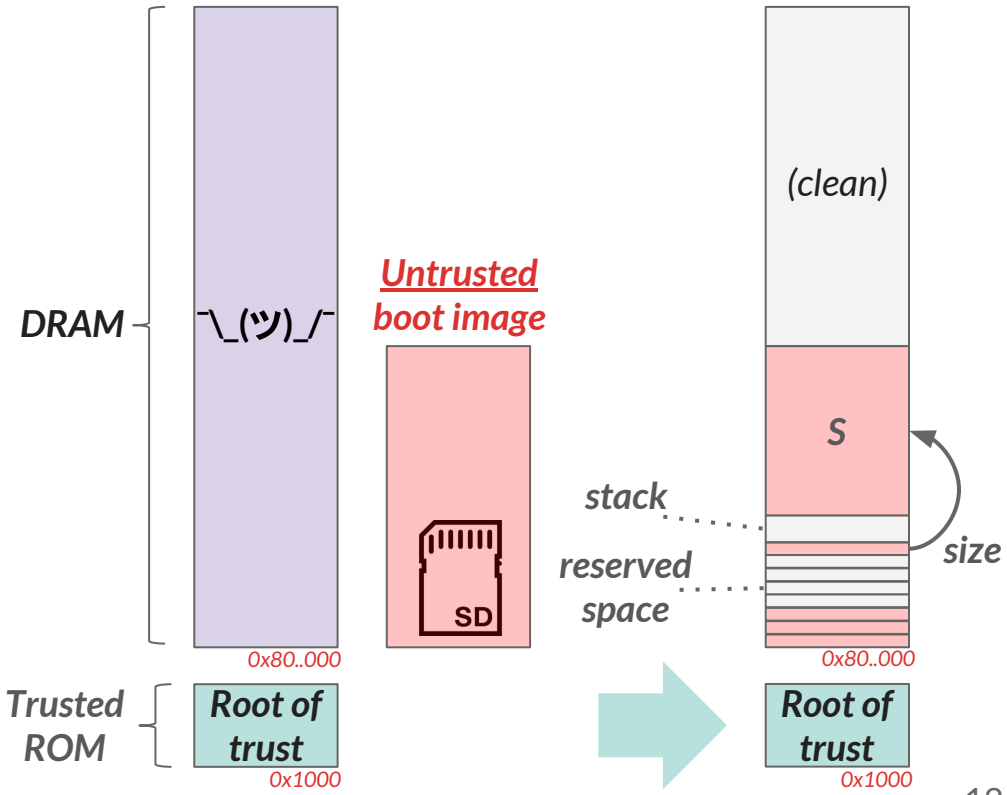
… secrets from previous boot

erasing GBs of DRAM is very slow!

2). **Load** an untrusted binary

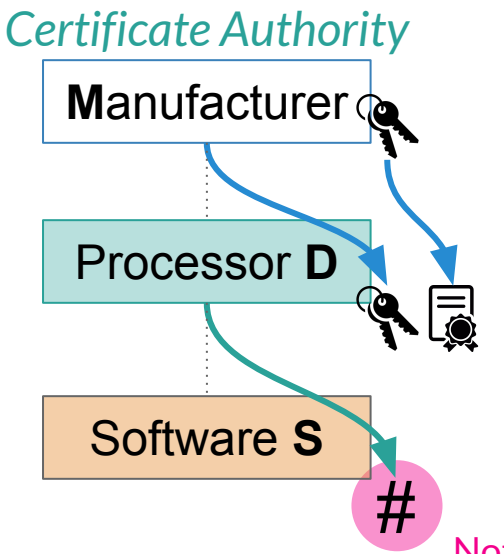
… assuming it has certain structure

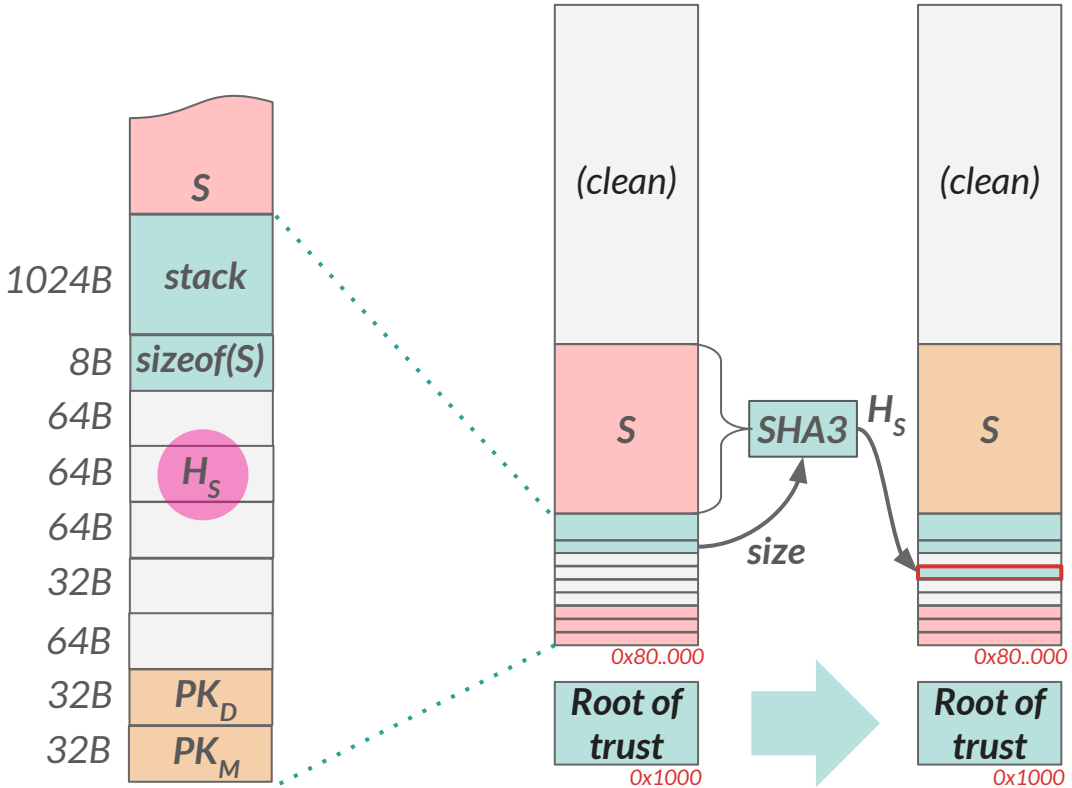
(size, reserved space for keys, etc.)



5: Root of trust measures S→H_S

Hash the binary **S** in memory → H_S



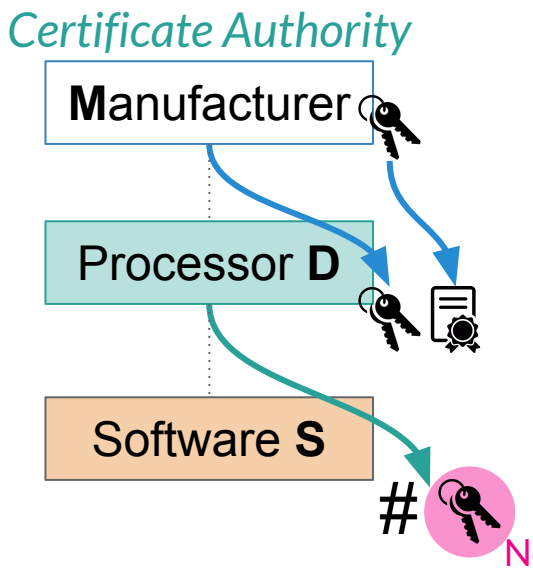


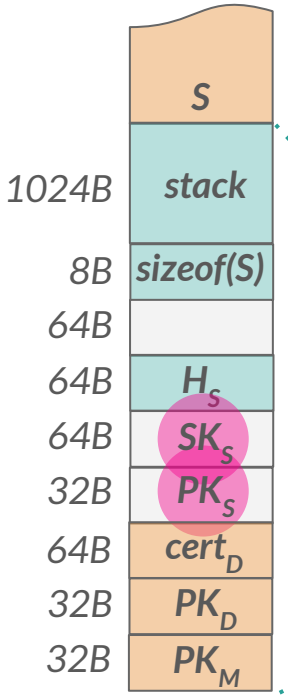
Note: can shrink the ROM by hashing and verifying the bootloader in RAM

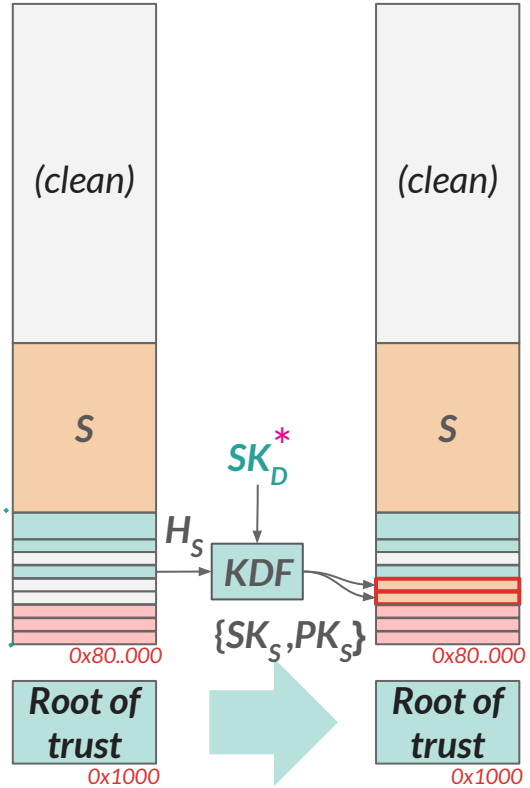
6: Root of trust endows S with unique keys

*D has access to SK_D, puts it on the stack.

$KDF(\{H_S, SK_D\}) \rightarrow \{SK_S, PK_S\}$





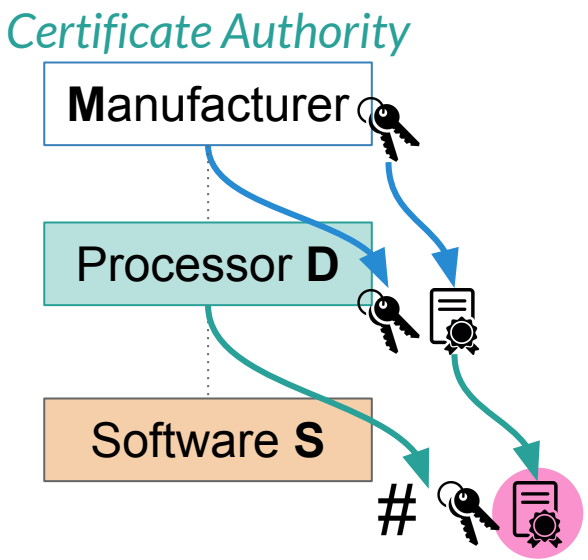


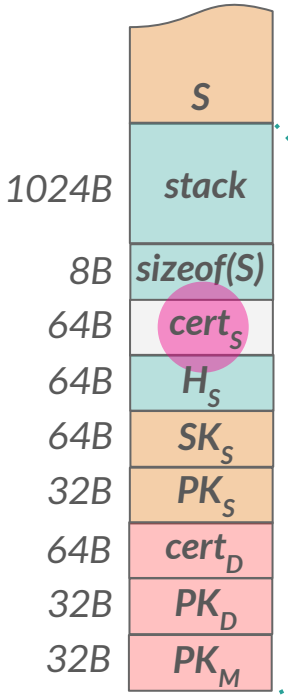
Note: can store encrypted SK_S in public, use SHA3(SK_D) to encrypt/decrypt

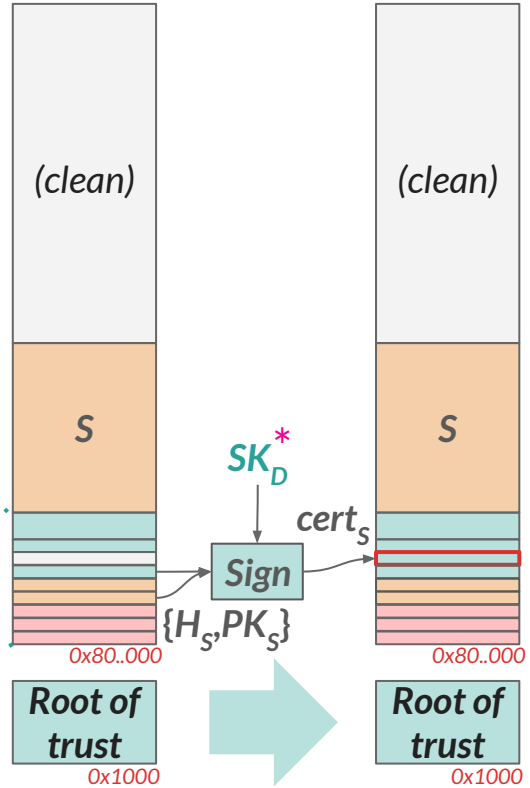
7: Root of trust certifies $\{H_S, PK_S\}$

*SK_D is on the stack.

$\text{Sign}(\{H_S, PK_S\}, SK_D) \rightarrow cert_S$





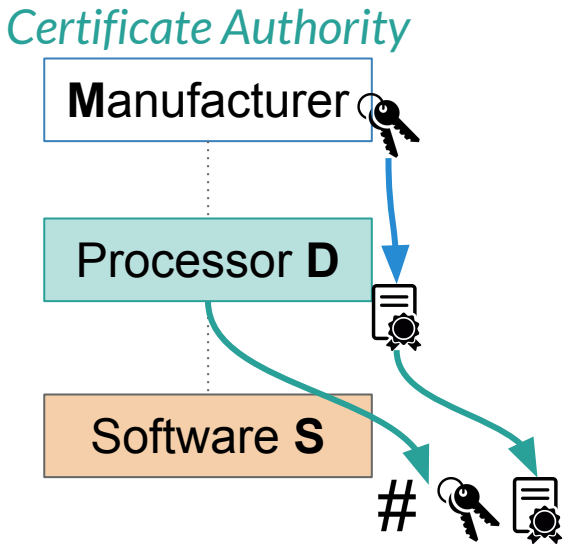


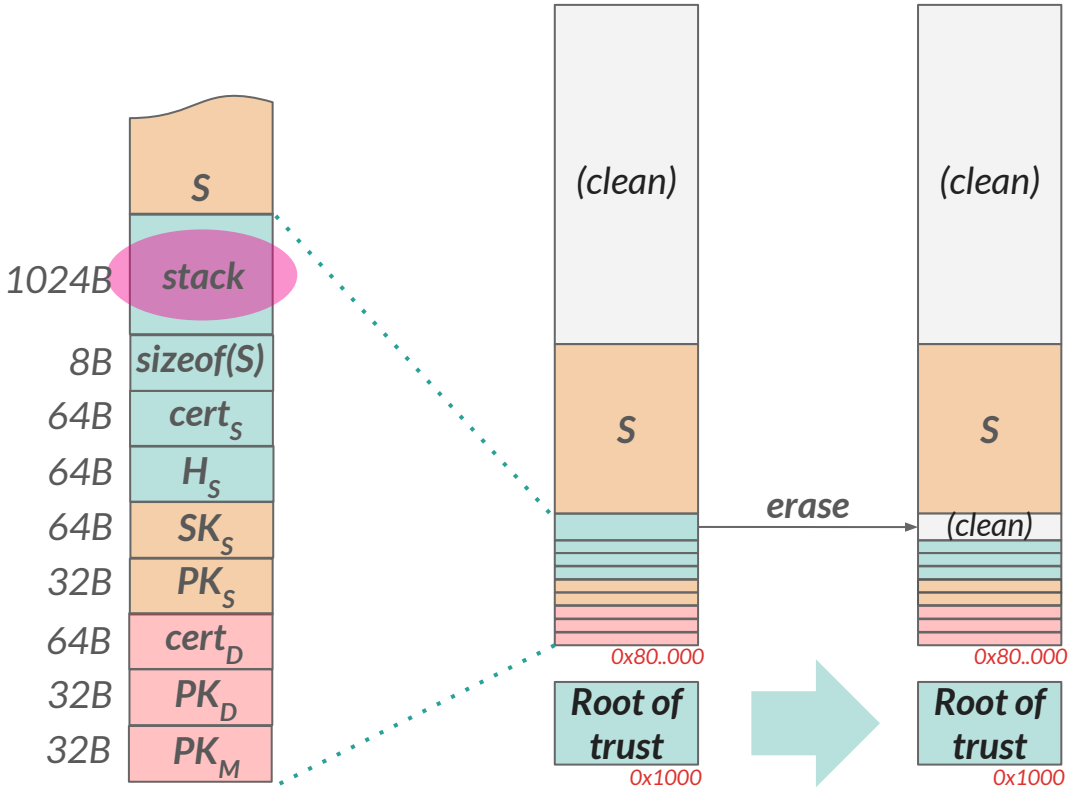
8: Root of trust cleans up and boots S

*Secrets are on the **stack***

*.. and in **µarchitecture***

*.. and **SK_D** must be hidden*



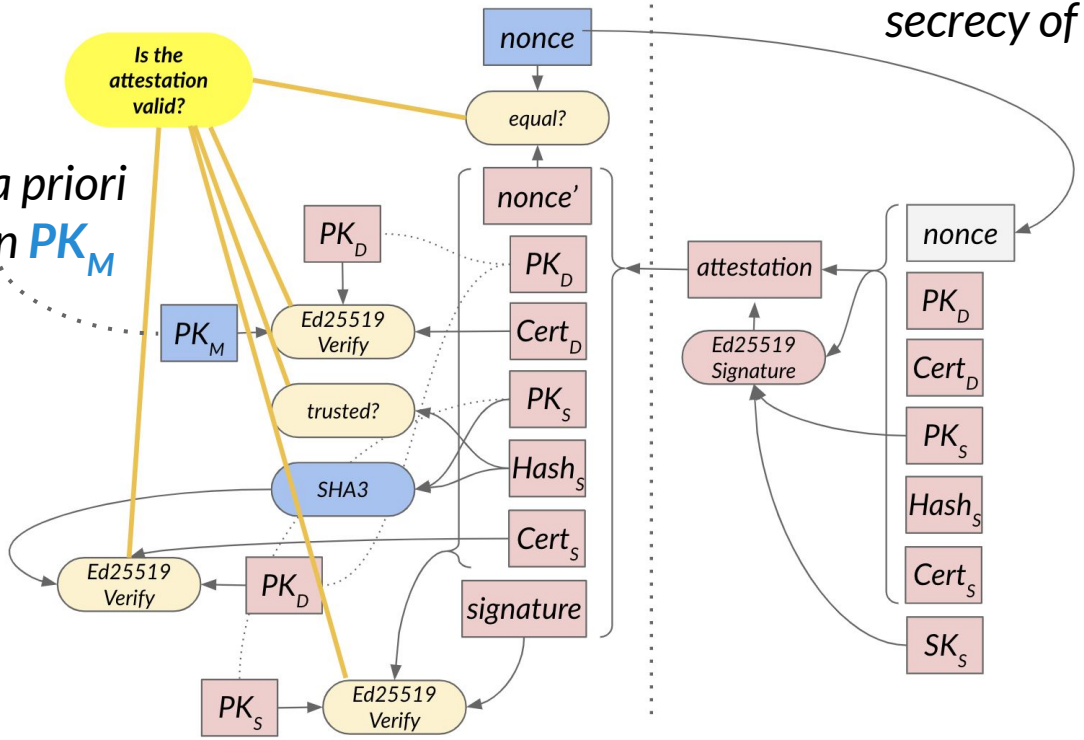


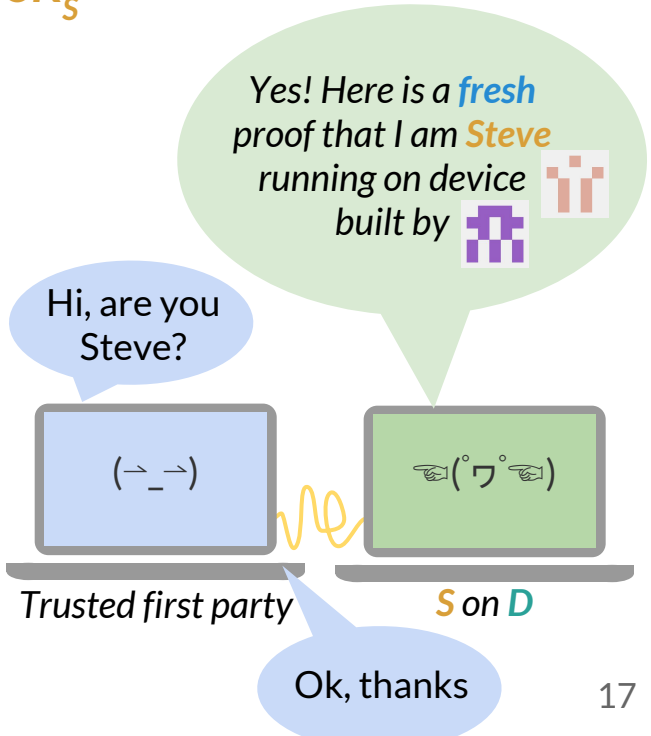
9: S performs attestation (1/4)

Trusted first party

S on D, trusted to maintain secrecy of SK_S

... has a priori trust in PK_M

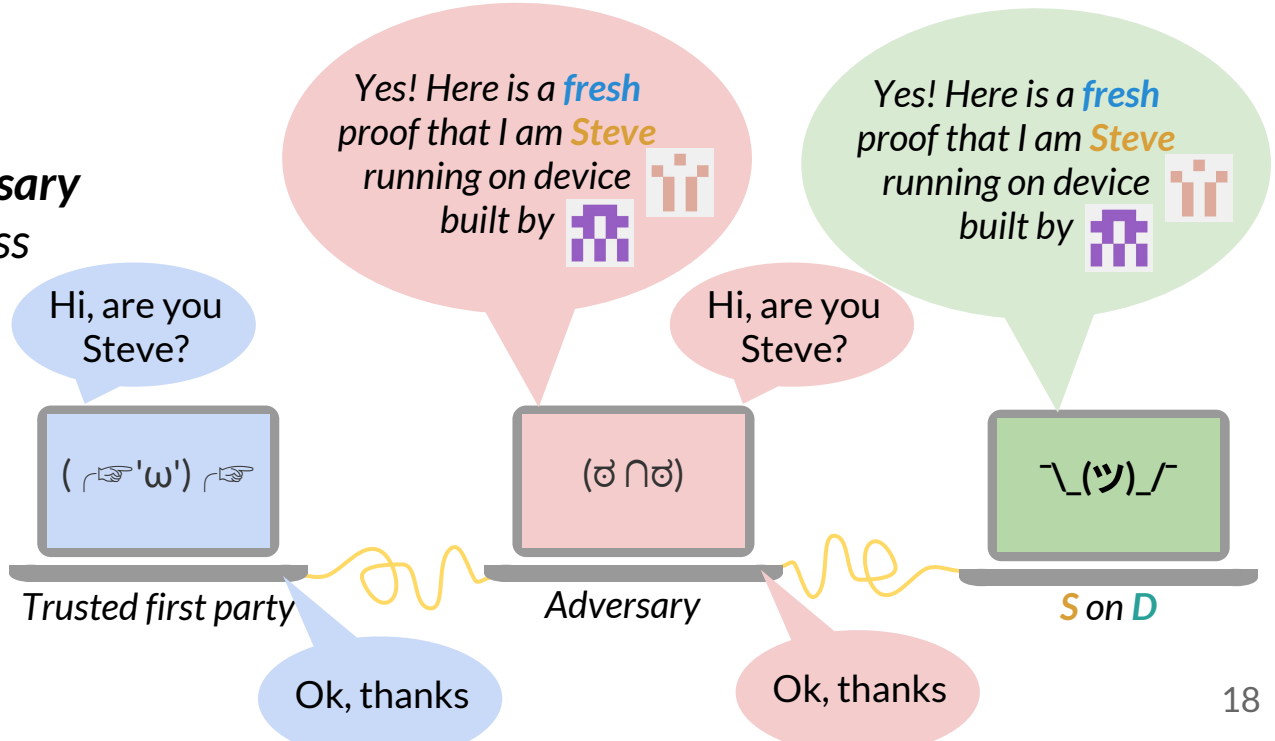




9: S performs attestation (2/4)

*An attestation only proves the sender **has access to** a genuine system.*

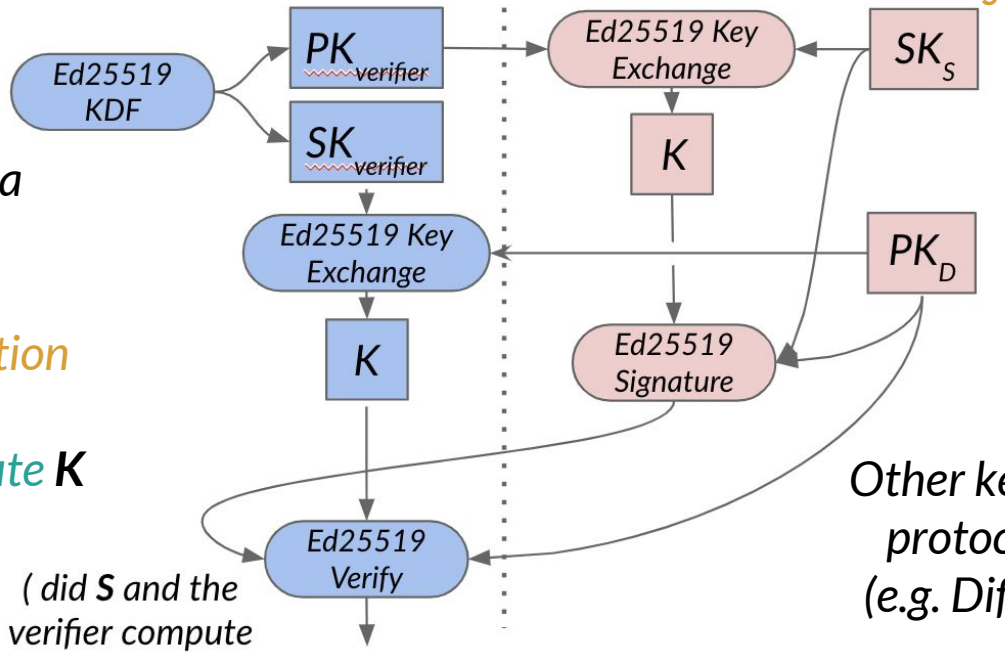
*A **man-in-the-middle adversary** (even a local one) may pass attestation*



9: S performs attestation (3/4)

Trusted first party

S on D, trusted to maintain secrecy of SK_S



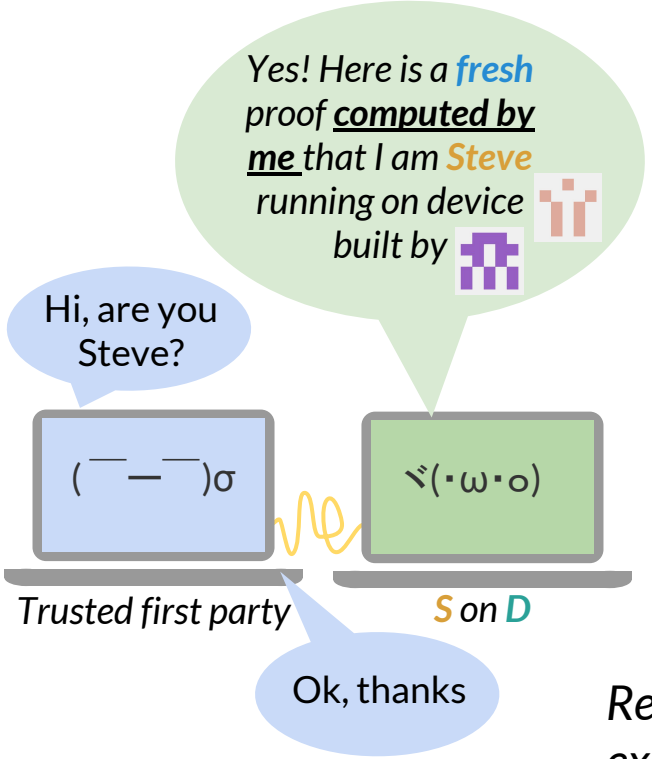
Key agreement *to defeat a man-in-middle; encrypt attestation with* ***K*** ***and*** *include* ***K*** *in the attestation*

The adversary cannot compute ***K***

(did ***S*** *and the verifier compute the same K?)*

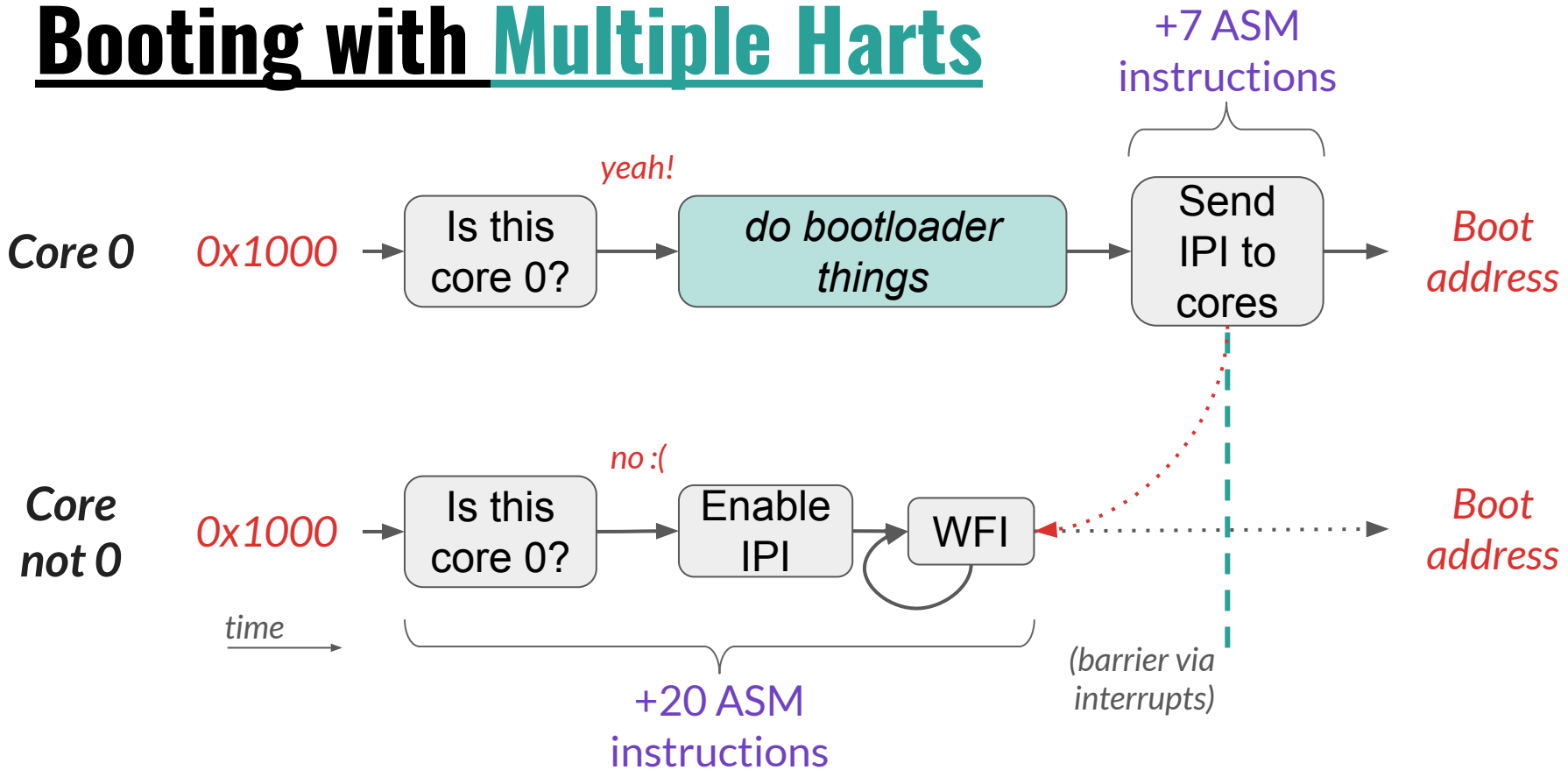
Other key agreement protocols exit too (e.g. Diffie Hellman)

We did it!



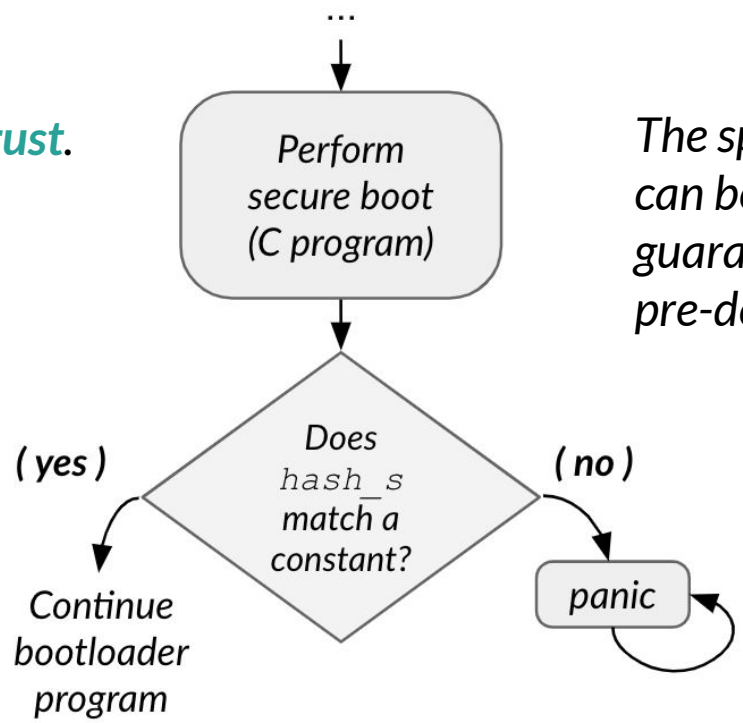
Read the Medium article for extensions to this boot protocol to make it more flexible

Booting with Multiple Harts



Trust S w/o 1st party: gatekeeping (1/2)

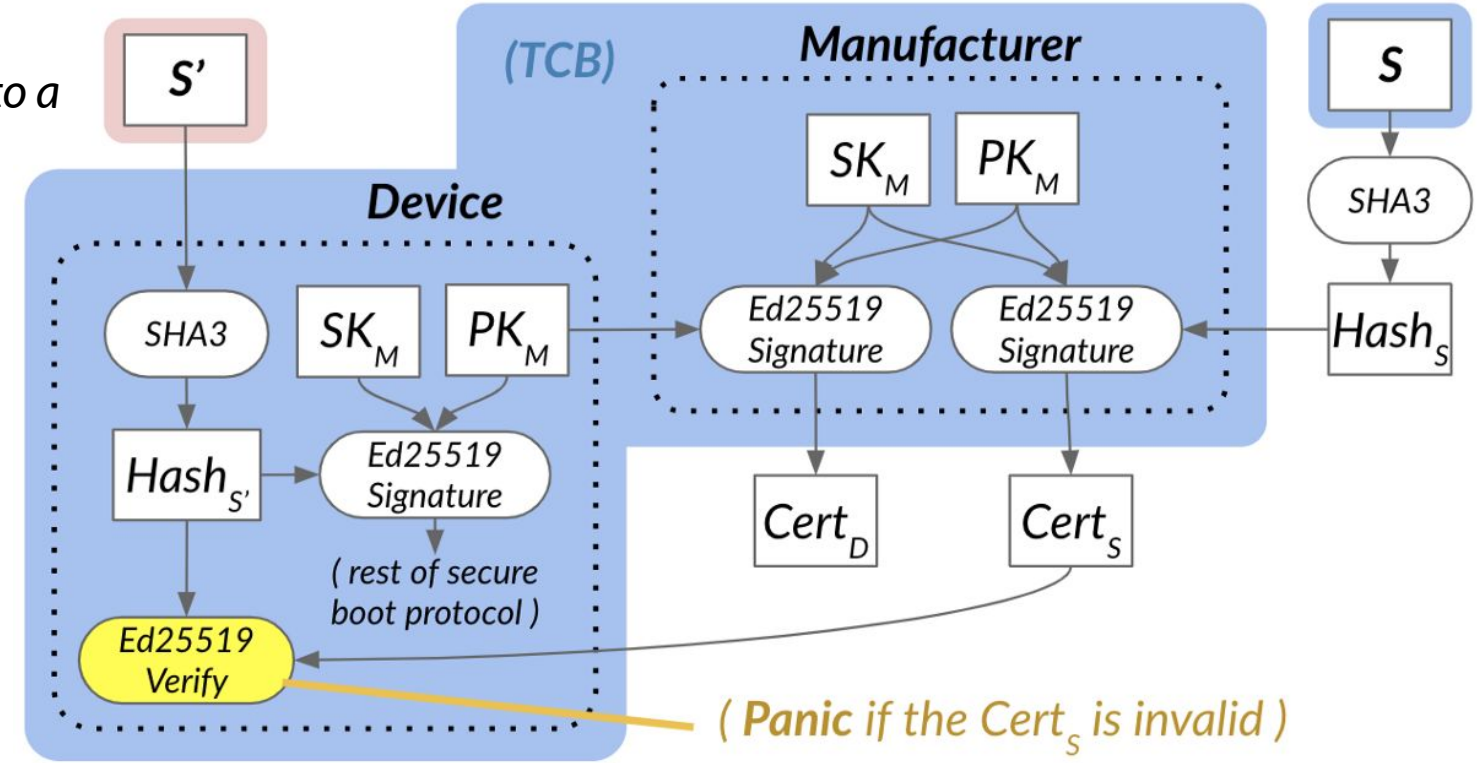
IDEA 1: *White-list the allowed S in the root of trust.*



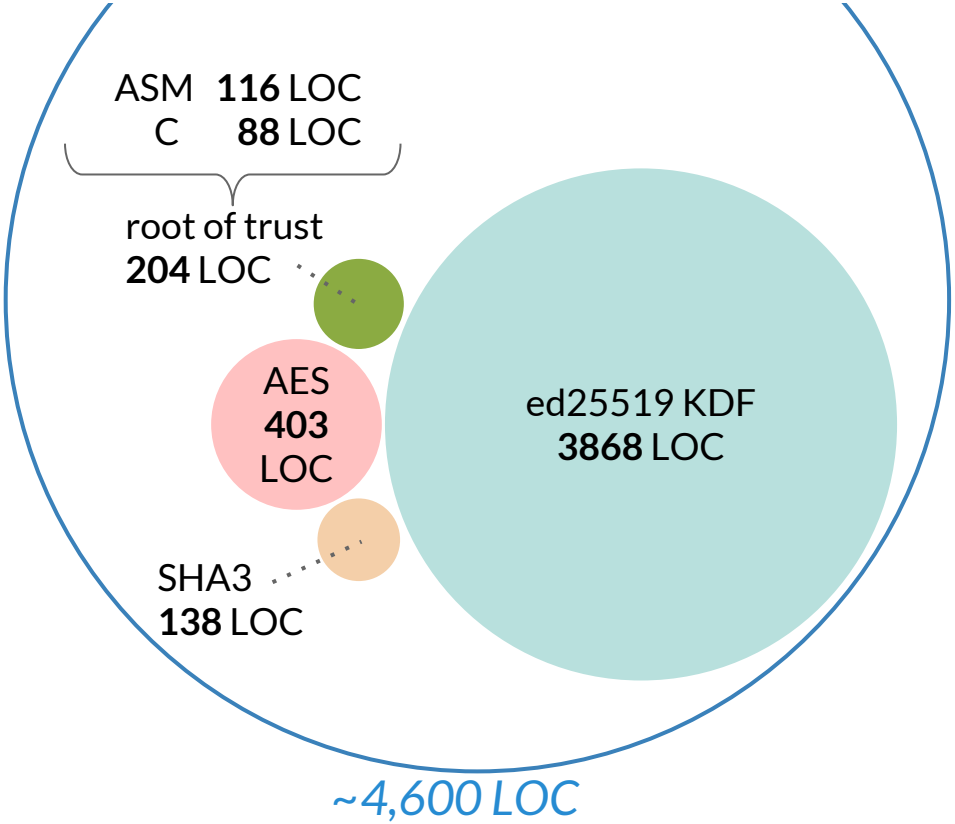
The space of software that can boot with an integrity guarantee from the RoT is pre-defined and ***immutable***

Trust S w/o 1st party: gatekeeping (2/2)

IDEA 2: *Delegate white-listing of S to a certificate.*



If a small TCB is the goal...



Not judging, but feature-packed bootloaders are huge:

- U-BOOT: ~2,000,000 LOC
- GRUB: ~326,000 LOC
- GRUB Legacy: ~43,000 LOC

The things we've swept under the rug...

- How can we trust S is keep SK_S secret?

Sanctum/Keystone,
or at least a PMP rule

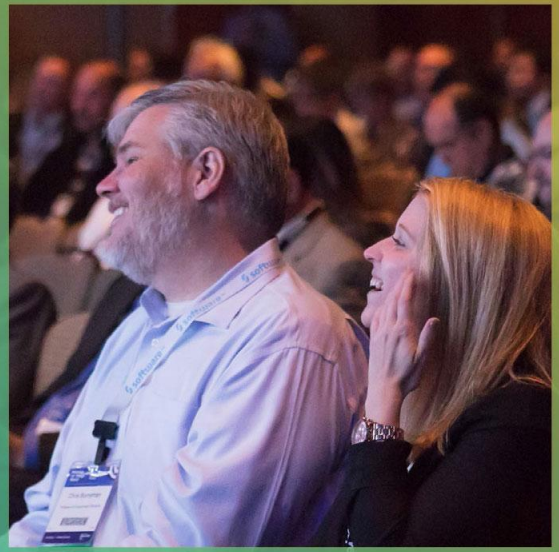
- Where does SK_D come from, exactly?

- What is our trust in the Manufacturer?

PUF-backed keys

RISC-V Summit

THANK YOU



https://tmt.knect365.com/risc-v-summit

@risc_v