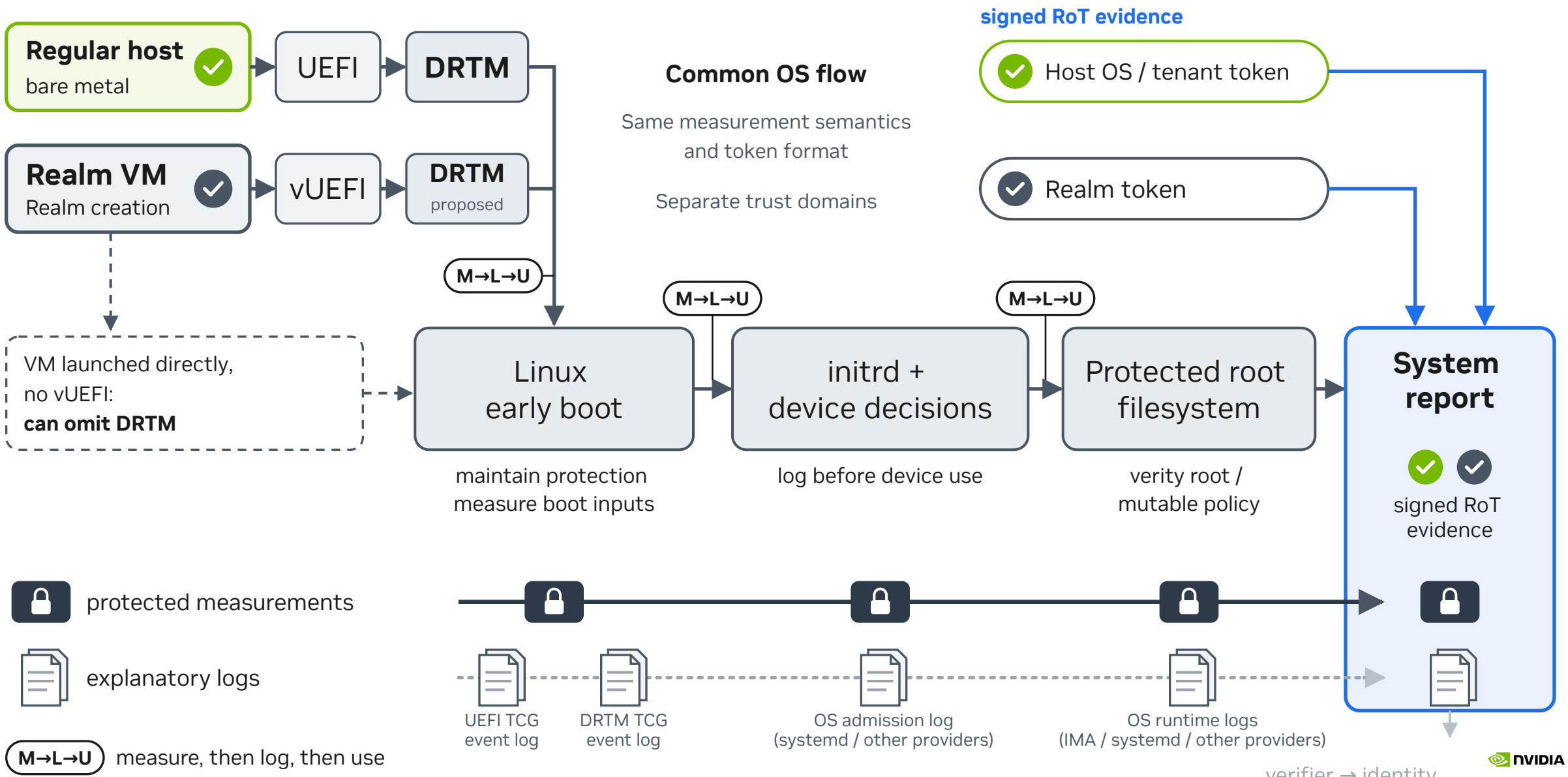
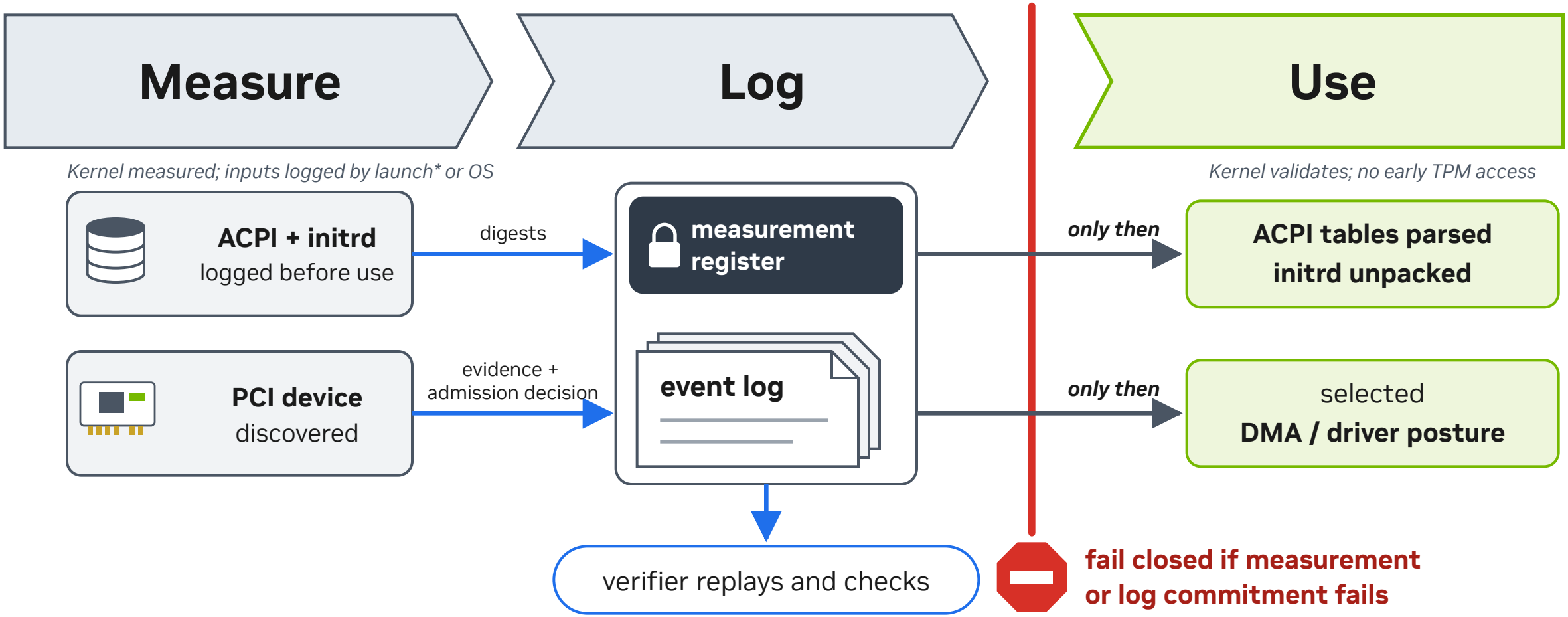
From protected launch to an attestable OS



Measure, log, then use

* Proposed: separate input measurements and TCB ACPI log entries.



Maintain protection from measurement through use.

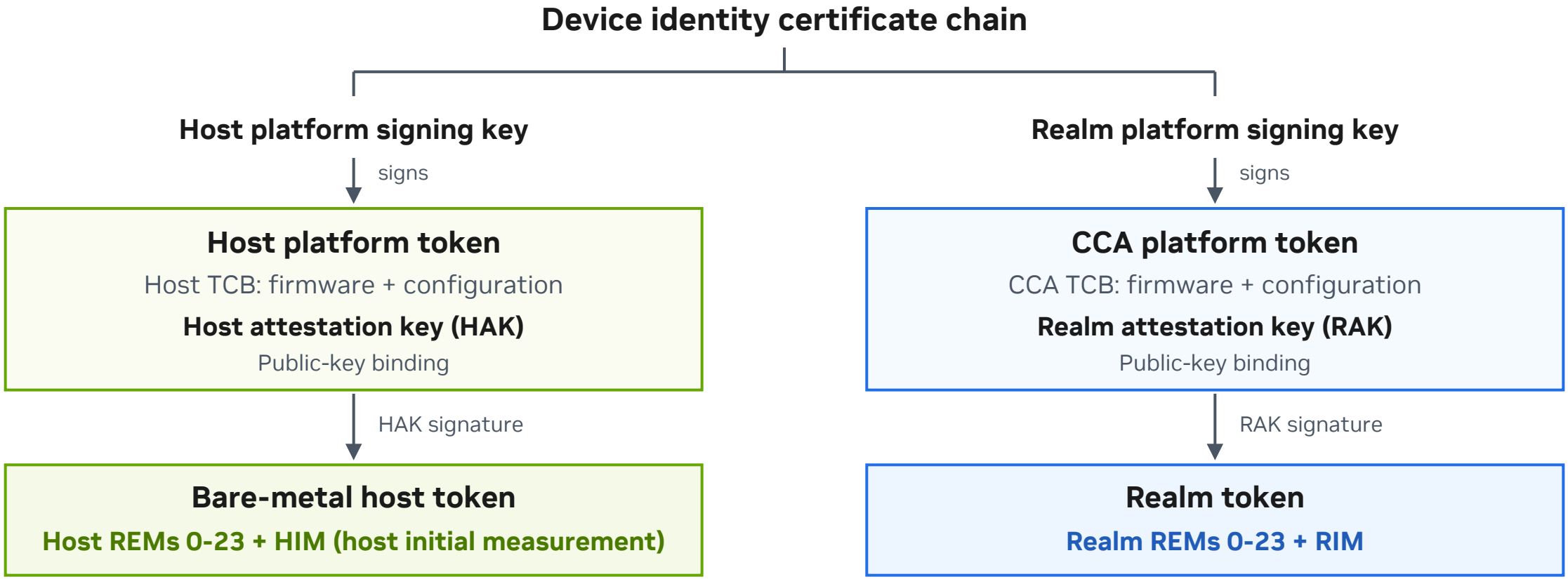
No committed record, no use.

NVIDIA

Proposed common attestation model

Expandable REMs: 24-register PCR-compatible baseline

Intended use: 1:1 TPM PCR mapping. Arm specs need not mandate register usage.
Guidance documents would help (e.g., SBSA-style guidance).



Common EAT format; separate banks. Host token, HAK and HIM are proposed.

Required: resettable REMs/PCRs with defined reset conditions and access control.

NVIDIA

Linux has names for the measurements

Example Linux conventions, reusable through 1:1 PCR-to-REM mapping.



UAPI PCR registry: existing Linux usage



Report service: systemd-report or another attestation service

OS conventions, not mandatory Arm assignments. DRTM has a separate launch profile.

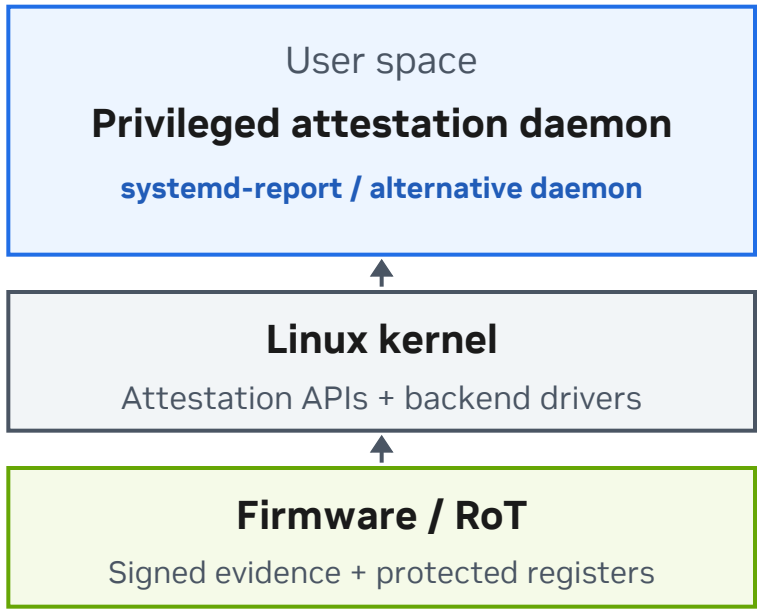
NVIDIA

A replaceable attestation report service

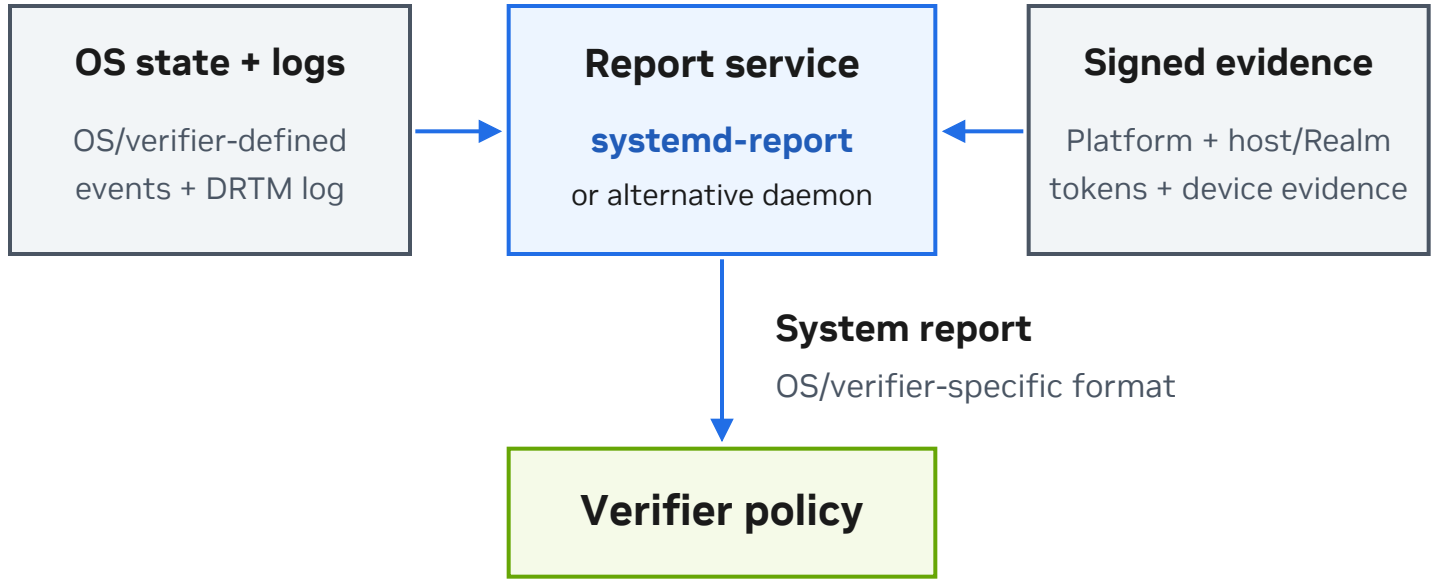
REMs can commit to any OS- or verifier-defined information

DRTM events retain their mandated format. Other measurement formats can be OS/verifier-defined.

Linux stack (proposed)



Report flow



Stable firmware tokens; OS measurement semantics evolve through event logs.

NVIDIA

Proposed DRTM extensions

Goal: defer kernel TPM/FIRME backends until initrd

Launch measures and logs; early kernel validates inputs without TPM/FIRME calls.

Proposed extension	Scope	Detail
1. Additional measurements	Memory ranges + event metadata; separate entries; protected handoff.	Slide 8
2. Non-TPM / FIRME	FIRME launch backend + discovery; normal-world bank and host evidence.	Slide 7
3. DRTM inside a Realm	RMM-mediated launch using REMs; protection + authenticated launch state.	Slide 10
4. ACPI logging	Launch logs TCB ACPI hashes; OS measures/logs remaining tables.	Slide 8

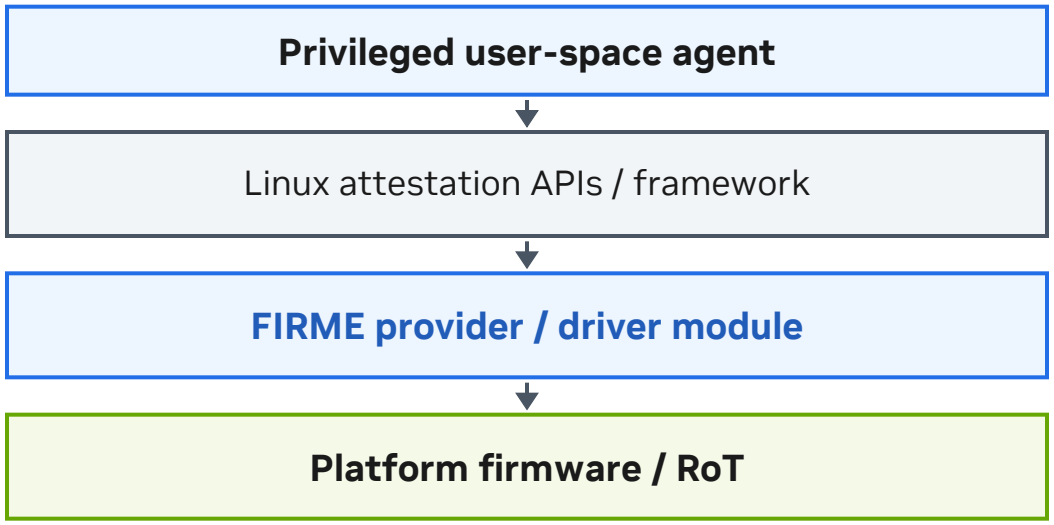
Discussion: scope, specification dependencies and implementation considerations.

NVIDIA

FIRME changes for regular-host attestation

Proposed boot flow and Linux attestation integration





FIRME changes + DRTM dependency

Host measurement bank: read / extend

Host token distinct from platform token

Reset / access rules aligned with DRTM profile

PCR-equivalent semantics (see slide 3)

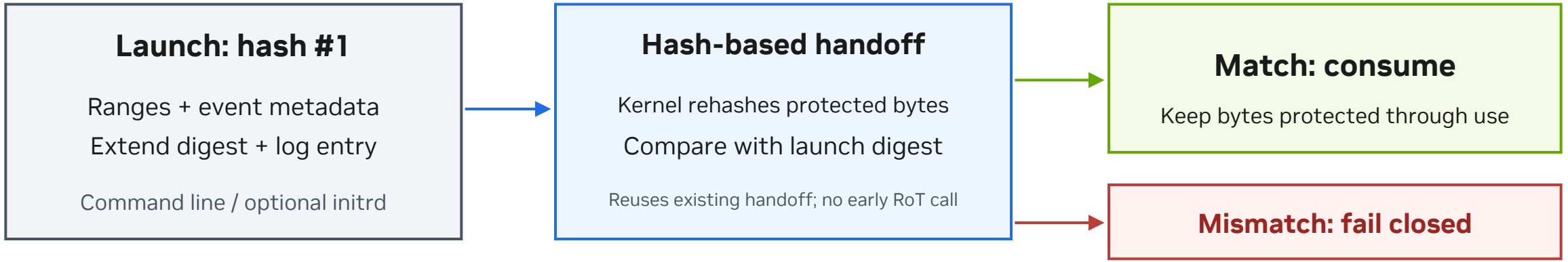
Target: no early kernel TPM/FIRME calls; initialize runtime backends from initrd.

NVIDIA

Additional measurements and ACPI logging

Record at launch, validate protected bytes before use

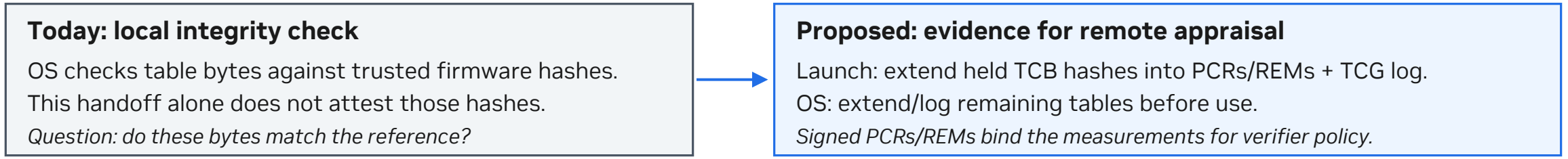
Ask #1: additional measured regions



Rehash can be omitted if launch measures the same bytes and protects them through use.

Bind addresses and lengths; exclude all untrusted writers. Objects need not be inside the DLME.

Ask #4: make the local ACPI integrity handoff remotely attestable



Why: expose ACPI inputs to verifier policy, without early kernel TPM/FIRME calls.

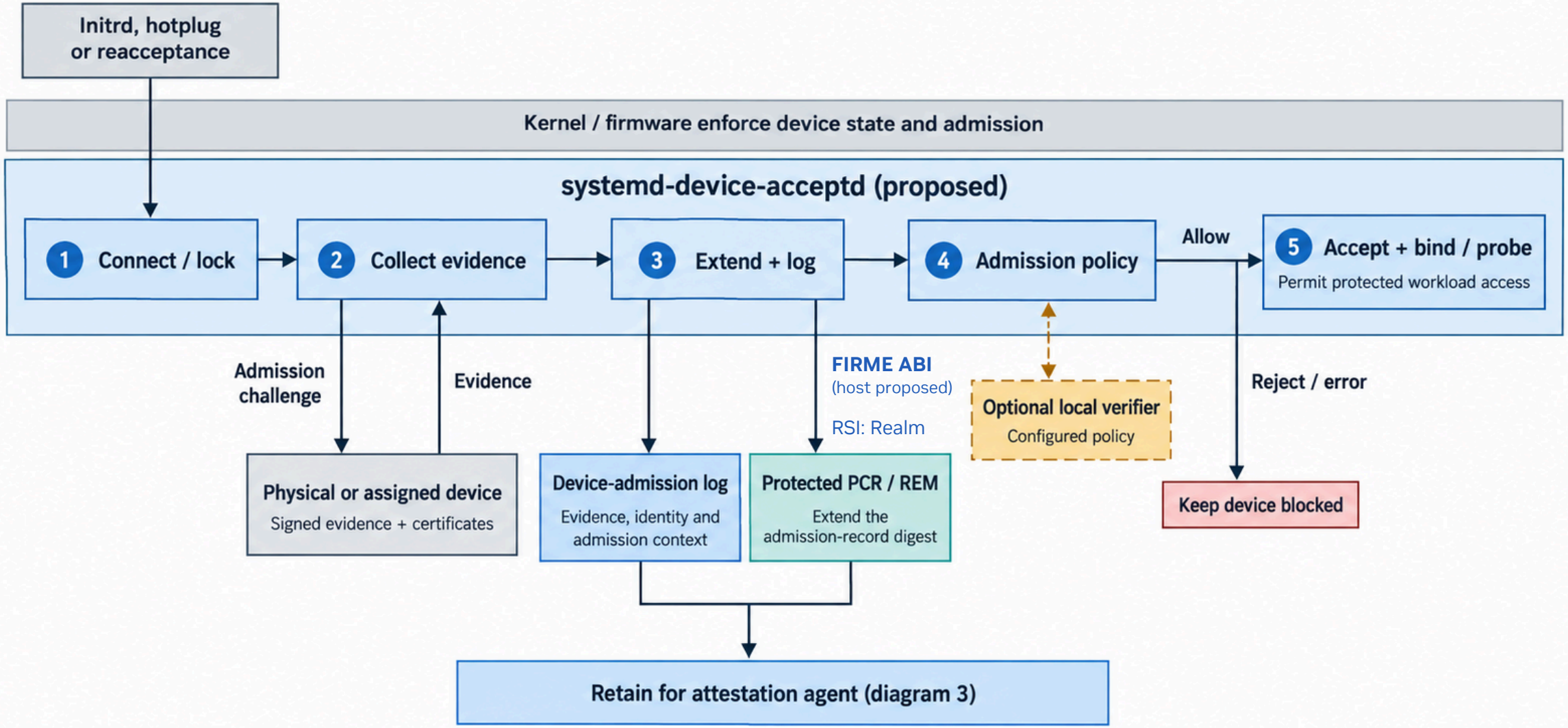
Logging supplied hashes does not remove trust in the firmware that supplied them.

NVIDIA

2. Device Acceptance

Target architecture

Unified ARM system architecture | Physical system or VM

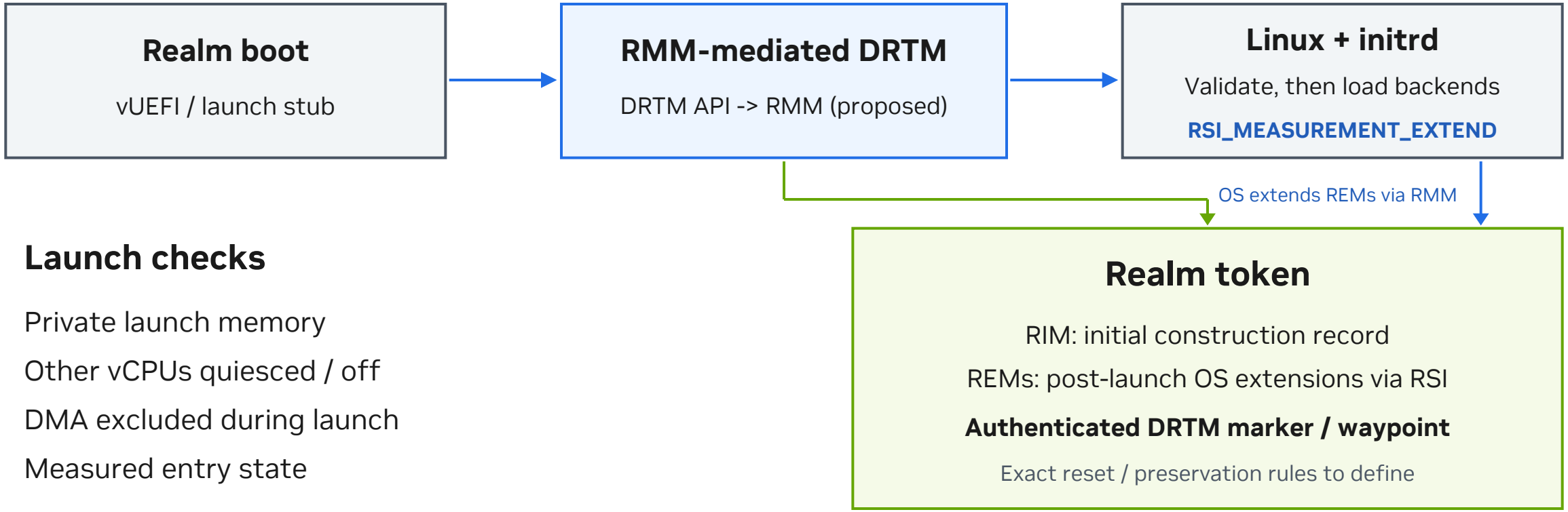


Record and extend before admitting the device.

Remote appraisal can follow admission; local policy determines whether to proceed.

DRTM inside a Realm VM

Proposed: reuse protected launch with an RMM backend



Launch checks

Private launch memory

Other vCPUs quiesced / off

DMA excluded during launch

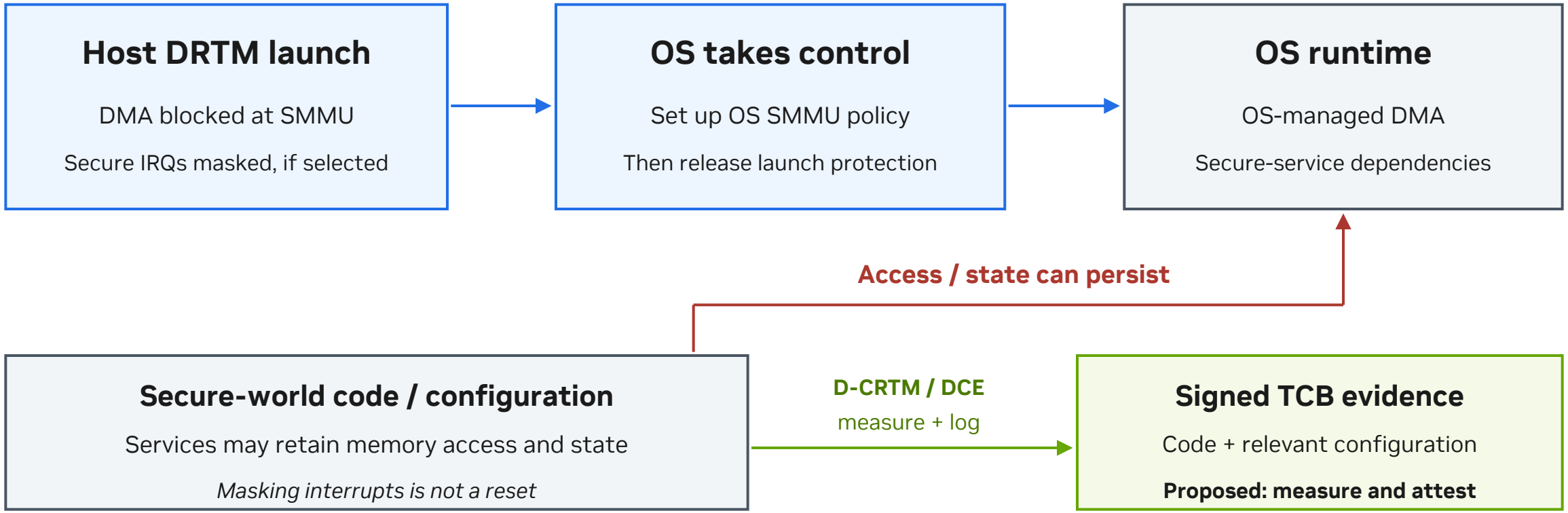
Measured entry state

Same OS flow; Realm-specific protection, keys and measurement bank.

Specify REM reset/access and launch evidence; retain RIM rather than replacing it.

NVIDIA

What remains trusted after DRTM?



Which secure-world code and configuration must be attested?

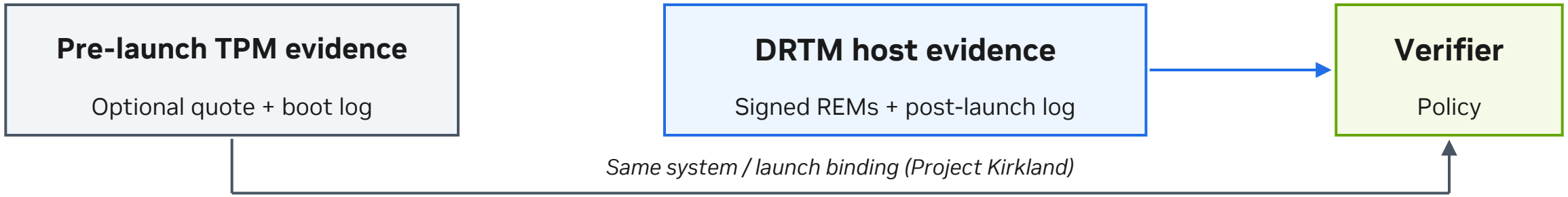
What state or access must be cleared before secure services resume?

When can Linux safely release launch-time DMA protection?

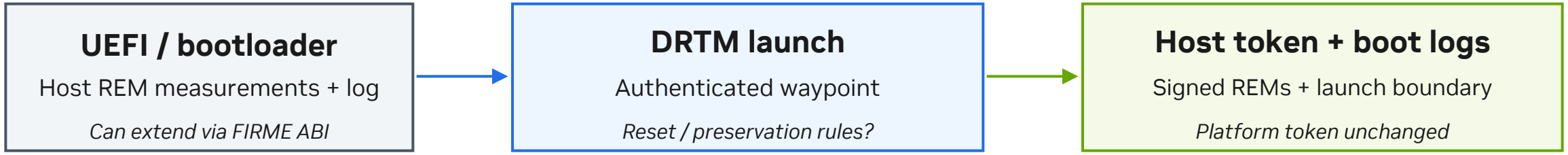
NVIDIA

Should attestation include the pre-launch history?

A. Separate TPM + DRTM evidence



B. Optional UEFI/bootloader history in host REMs (proposed)



Should pre-launch evidence remain optional?

TPM evidence or host REMs: how do we bind the history to this launch?

How do we preserve DRTM resets without requiring trust in the full boot chain?