European Heterogeneous Cloud / Edge infrastructure for Next Generation Hybrid Services

Nick Kossifidis - FORTH

Self-Signed Silicon: Who Vouches for the Chip When Providers Make Their Own?

Webinar hosted by HiPEAC [https://hipeac.net/webinars/]

October 29th, 2026

The old world: three parties, one anchor

- Every hosted workload has three parties with different stakes:
  - Service / host provider - owns and runs the machine
  - Payload provider - the service, code or model that runs on it
  - Data provider - the end user whose data is processed
- Each must trust hardware it does not physically control
- In the x86 era that trust was anchored by a third party: Intel or AMD

The operator ran the machine, but did not mint its identity or its keys.

What x86 silicon vouched for

- Tracked, versioned CPU / SoC identities issued by the vendor
- Firmware TPM (fTPM) and measured boot rooted in vendor keys
- Memory encryption and confidential computing: Intel SGX / TDX, AMD SEV-SNP
- All endorsed by a party that did not operate the data center
- The provider could not trivially forge or bypass it on its own

Independence came for free: the chip vendor was not the cloud operator.

When the operator makes the chip

- Hyperscalers already design their own silicon:
  - AWS (Graviton, Nitro), Microsoft (Cobalt, Maia), Google (Axion, Titan), Meta (MTIA) - mostly Arm today
- Open ecosystems lower the barrier: RISC-V, OCP, Caliptra - many more will follow
- The third party disappears: the operator now mints identity, RoT keys and attestation evidence
- What weakens, in order:
  - Integrity - is the chip running the firmware it claims?
  - Authenticity - identity and signatures are now the operator's
  - Confidentiality - memory-encryption keys are held by the operator

Owning the chip changes who you trust, not whether you can check.

A restaurant and its health inspector

Before: inspector from Intel / AMD

An independent inspector vouched for the kitchen. You trusted the food because someone who did not own the restaurant signed the report.

You could rely on the certificate.

Now: the restaurant's own inspector

Same kitchen, same report - but the inspector now works for the owner. The certificate no longer proves anything to you.

Same food, different signature.

The takeaway, without the jargon

Sovereignty decides who you trust - not whether you can check.

What the hardware actually gives you

Mechanism	Defends against	What you really get
Secure / measured boot	Tampered or unknown firmware	Best-effort; meaningful only if the RoT is genuine
Isolation - IOMMU / IOPMP, TEE	Co-tenants, rogue devices	Best-effort; bugs and side-channels keep surfacing
Memory encryption - TME / SEV / TDX	Bus snooping, cold boot	Partial; replay and physical mostly out of scope
Attestation & identity - RoT keys, SPDM	A host lying about its state	Only as good as who holds the key

Best-effort mechanisms we choose to trust - not proofs. The real question: is the mechanism even there?

Trusting the effort vs. trusting a claim

	Third-party silicon	Provider-made silicon (may be emulated)
Is the mechanism really present?	Yes - built by an independent vendor	Maybe not - an emulator or debug part can fake the evidence
If it is bypassed	Needs a physical, per-machine attack	**One policy switch; every guarantee void at once**
What you ultimately trust	The vendor's best effort, independently attested	The operator's word about its own chip
Who holds the pieces	Silicon, RoT and operator are	One party owns silicon + RoT +

The fix isn't a stronger mechanism - it's a smaller, independently-vouched one: shrink what must be trusted.

Caliptra: transparency, not independence

What it gives

Open root-of-trust IP with DICE identity and signed measurements. OCP S.A.F.E. audits and one design across vendors. OpenPRoT platform firmware on top.

Its stated goals: transparency, consistency.

What it leaves open

Who endorses the device key. Whether the silicon matches the RTL. How the root of trust is bound to the compute it describes. The isolation mechanism itself.

Identity and measurement, not independence.

Where independence has to come from

Endorsement · Verification · Reference values must sit outside the party that operates the hardware.

Who endorses the device identity?

- x86: identity baked into the SoC, guaranteed by Intel / AMD - evidence about a real CPU
- Provider-made: identity can be forged; an emulated or debug environment can claim guarantees it skips
- A path: derive the device ID from Caliptra's UDS (via a KDF) - sound only if:
  - the UDS stays secret from everyone, including the operator
  - a neutral party (the fab) endorses it:
    - “A chip I made”
    - “With vanilla Caliptra subsystem, version X”
- Open problem: key custody plus an endorsement protocol a tenant can verify

A self-endorsed identity is not an identity - it needs a voucher from outside the operator.

The policy gap: no independent voucher

- Cloud compliance certifies processes and products - not a per-device, independently endorsed identity
- None requires the party running the hardware to differ from the party vouching for it
- Where such a rule could live:
  - Finance: DORA - oversight of critical cloud providers to banks
  - EU: EUCS, Cyber Resilience Act, NIS2. US: FedRAMP
- Apple PCC shows the pattern: on its own silicon, transparency + non-targetability; on Google Cloud, independence via a hardware ledger +

Whoever runs the hardware must not be the only one who vouches for it.

HIGHER: a security engine subsystem

- A dedicated security engine: Caliptra RoT plus an MCU for critical operations
- Secure boot and attestation services for the platform
- Also handles power management, clock control, and static isolation via IOPMP
- The anchor the rest of our mechanisms build on

One auditable subsystem owns identity, measurement and isolation.

NetBoot: trust from device to network

- Bring the whole system up from network-delivered images, not local flash
- Why network boot:
  - Maintenance: one managed image set, not per-device flash to service
  - Cost: no 2x redundant flash, and the update-rate / wear limits it imposes
  - Cleaner: no on-device firmware structures; nothing persistent to implant; fresh state each boot
  - Prototyping: works on FPGAs with no local storage
- The local Caliptra RoT verifies every image before it runs

If only the DC can sign legitimate images, the device can trust its environment.

NetBoot: trust from network to device

- The other direction: the infrastructure must trust the device before handing it images
- We explore device identity and onboarding protocols with IEEE 802.1X port-based auth, before boot
- Hard: this lives in a boot stage that wants to be the bootrom
  - keep it small, simple, auditable; minimal attack surface
  - existing onboarding protocols do not fit this constrained environment
- Work in progress - sound only under the key-custody and neutral-endorser assumptions

Mutual trust before the first image - a bootrom-sized problem.

What we claim, what we don't, what's open

- We claim:
  - The trust model is shifting, and it is rarely discussed
  - Caliptra and OCP improve transparency; network boot + a log keeps hosts clean and deviations detectable
- We do not claim:
  - A complete solution, or that RISC-V / sovereign ownership removes the problem
- Open questions:
  - Independent endorsement across jurisdictions; checking fabricated silicon against open RTL; tenant-run verifiers

Sovereignty needs verifiability, not just ownership. Collaborate via HIGHER, RISC-V, OCP.

Thank you for your attention.



https://higher-project.eu

Disclaimer: "Funded by the European Union under Grant Agreement 101189612. Views and opinions expressed are however those of the author(s) only and do not necessarily those of the European Union or the European Health and Digital Executive Agency (HaDEA). Neither the European Union nor the granting authority can be held responsible for them."

Backup: physical attacks on TEEs

- Memory encryption and isolation on x86 TEEs are best-effort; physical attacks keep finding gaps
- 2025: Battering RAM and WireTap (DDR-bus interposers); TEE.Fail extracted attestation keys on DDR5
- Practical and cheap - TEE.Fail used a sub-$1,000 logic analyzer; cache-bypass / replay aided earlier ones
- Vendors (Intel TDX, AMD SEV-SNP) state physical attacks are out of scope
- Still narrow: physical access, specific victim code, one machine at a time - they break a mechanism
- A provider that owns the chip can skip the mechanism instead - fleet-

Breaking a mechanism is bespoke and physical; skipping it is a policy switch.

Backup: DPE vs a firmware TPM

DPE (Caliptra)

DICE Protection Environment plus secure-boot measure / stash / extend. Purpose-built: small spec, simpler firmware.

Smaller attack surface.

Firmware TPM

TPM 2.0 is a large, general-purpose spec (four parts, 1000+ pages); much of it is rarely exercised in server attestation.

More surface, more legacy modes.

The point

DPE is an evolution of the TPM / fTPM idea for a root of trust - less to implement, less to get wrong.

References & source material

- Remote attestation (RATS architecture) - IETF RFC 9334
- Caliptra - github.com/chipsalliance/caliptra-sw; Microsoft Azure blog (2022)
- OCP S.A.F.E. / S.O.L.I.D. - opencompute.org/projects/security
- DICE DPE - Trusted Computing Group (trustedcomputinggroup.org)
- Apple Private Cloud Compute - security.apple.com; Google Cloud expansion, InfoQ (2026)
- TEE physical attacks - Battering RAM, WireTap, TEE.Fail (2025)
- EU policy - DORA; EUCS (ENISA); Cyber Resilience Act; NIS2
- Flashless / network boot prior art - HPE, FOSDEM 2025
- HIGHER project - higher-project.eu