The Trusted Chip Is Not the Trusted Signal

How Signal Veracity Changes Hardware Partitioning in Consequential Systems

Dana Xiadani
Founder & Chief Architect, Living Cipher
dana@livingcipher.com | www.livingcipher.com
Living Cipher Analysis 002 | August 2026 | Version 1.0 | DOI: 10.5281/zenodo.22019132

Central claim
A trusted processor can execute authorized computation perfectly on an observation that should never have been trusted.

Abstract

Hardware roots of trust, secure boot, measured state, attestation, and protected communication can establish strong evidence about the identity and operating state of a computing platform. They do not, by themselves, establish that a physical observation entering that platform remains coupled to the claimed source or event, is fresh enough for the intended decision, or satisfies application-specific admission requirements. In sensor-driven medical, defense, industrial, and autonomous systems, that distinction changes architecture. Observation-evidence requirements can determine which properties must be captured or bound near acquisition, which functions require protected execution or non-bypassable enforcement, which evidence may be transported downstream, which logic should remain programmable while the evidence model evolves, and which stable trust invariants may eventually deserve dedicated silicon. This paper introduces evidentiary locality as a hardware-partitioning pressure and proposes a signal-to-substrate framework in which the consequential claim defines the required evidence; the evidence requirements define the trust boundary; the trust boundary constrains partitioning; and partitioning, together with economics, determines substrate. The result is a broader rule for consequential compute: performance determines where computation is efficient; trust determines where computation remains valid.

Keywords: hardware trust; signal veracity; trusted sensing; hardware/software partitioning; remote attestation; FPGA; ASIC; consequential systems

1. The Trust Boundary Begins Too Late

Modern trusted-computing architectures have become increasingly capable of answering questions about the machine: Which device is this? What firmware started? What measurements were recorded? Which keys are protected? Is the execution environment in an authorized state? Can the device produce evidence that another party can appraise? Those are necessary questions. In many consequential systems, however, they are not the first trust questions the architecture must answer.

A sensor-driven system also has to ask whether the observation entering the trusted machine supports the physical claim being made about it. A signed packet can contain a replay. An authenticated device can forward a substituted signal. An approved model can execute correctly on a stale observation. A hardware root of trust can establish machine identity while the physical source has already been decoupled from the data path.

This paper uses signal veracity in a deliberately bounded sense. It does not mean metaphysical truth, clinical truth, or proof that every sensor value is correct. It means evidence bearing on whether an observed signal remains sufficiently coupled to the claimed physical source or event under the deployed threat model, including resistance to replay, substitution, injection, or source decoupling. Freshness, provenance, device/session binding, and signal quality are related but distinct evidence attributes that may also be required for admission. A noisy but source-coupled observation may be low quality; a pristine replay may be high quality and still fail the source-coupling requirement.

That distinction matters because downstream cryptography can preserve the integrity of whatever it receives. If the wrong object crosses the trust boundary, a system can preserve the integrity of a false semantic premise with exceptional rigor. The architectural question is therefore not whether conventional attestation is inadequate. It is whether the trust boundary has been placed far enough upstream for the claim the system intends to make.

Scope and contribution. This paper does not propose a new cryptographic primitive, a universal liveness detector, or a claim that trusted-sensing research has ignored capture integrity. Prior work has shown that live sensor capture and data-capture rules can themselves be made attestable.[1] The contribution here is architectural: treat observation-evidence requirements as a first-order partitioning variable and trace their consequences across ingress placement,

programmability, memory ownership, accelerator and chiplet interfaces, failure behavior, and eventual substrate commitment. This is an architecture thesis, not empirical validation of any particular source-veracity mechanism.

Threat-model boundary. This analysis considers attacks or failures that can break the link between a claimed physical source or event and the data admitted for consequential processing, including replay, substitution, injection, source decoupling, stale capture, and loss of custody where relevant. It does not assert that architecture alone establishes clinical truth, sensor calibration, semantic correctness, or model validity; those remain separate validation burdens.

2. Trusted Device Does Not Imply Trusted Observation

Remote-attestation architectures correctly separate evidence generation, appraisal, and the relying party's decision. RFC 9334 deliberately keeps claim content and attesting-environment implementation broad; it defines Evidence about a Target Environment, requires reliable collection and secure association with that environment, and treats freshness and appraisal policy as explicit architectural concerns.[2] That architecture can be applied to many kinds of targets, but it does not itself define a physical-source veracity model. Platform roots of trust such as Caliptra focus on identity, measurement, measured boot, and attestation of the SoC and its security-relevant state.[3] SPDM provides standardized mechanisms for authentication, measurements, and session key exchange, and is designed to support confidentiality- and integrity-protected data communication.[4]

These mechanisms are powerful because they make platform and component claims harder to counterfeit or silently modify. They do not claim to establish the physical truth of every sensor observation presented to the machine. That is not a defect in their scope. It is a boundary condition for system architects.

Three claims should therefore remain separate in architecture reviews. Observation evidence asks whether the input satisfies the application's source-coupling and related freshness, provenance, and admission requirements. Machine evidence asks whether the device and execution environment present the identity, measurements, and protected state required by policy. Model or algorithm validation asks whether the computation is fit for its intended purpose. None of these claims subsumes the other two. A trustworthy platform can run an invalid model; a valid model can receive a replayed observation; a source-coupled observation can enter a compromised machine. Observation evidence should therefore remain a structured set of evidence claims and state, not collapse into a single scalar 'trust score.'

> **Conceptual admission predicate**
> Consequential compute or actuation is admissible only when observation evidence and machine evidence satisfy the applicable policy, including any requirement on authorized model or algorithm state. Strong evidence in one domain cannot silently substitute for missing evidence in another.

Exhibit 1. Two evidentiary domains and the binding between them

Physical observation	Acquisition boundary	Source-coupling evidence	Appraisal / admission	Consequential compute
Claimed source or event	Device / session / freshness context	Protected evidence + binding	Policy decision + appraised machine / compute state	Provenance-bound result
Required binding = admitted observation evidence ↔ appraised machine / compute state ↔ policy ↔ result provenance				

The important architectural move is to stop treating these domains as interchangeable. Machine evidence says something about the computing environment. Observation evidence says something about the physical premise the machine was asked to accept. A consequential result may need both. The system should therefore preserve a verifiable relationship among the admitted observation, the evidence and policy supporting admission, the execution state under which computation occurred, and the resulting output provenance.

3. The Cryptographically Immaculate Lie

Consider a physically sourced input that is captured once, replayed later, and delivered through an authenticated channel to a device with secure boot and a valid attestation chain. The device may be exactly what it claims to be. The software may be approved. The packet may be intact. A downstream timestamp may be fresh. The computation may

be deterministic and correct. Yet the observation itself may no longer be contemporaneous with the physical state the system believes it is evaluating.

This is the cryptographically immaculate lie: the cryptographic envelope can be internally consistent while the semantic premise at the front of the chain is wrong. The failure is not that cryptography lied. The failure is that architecture asked cryptography to certify a property it was never positioned to observe.

Signing at the sensor or acquisition gateway can materially improve provenance and transport integrity, but a signature alone does not establish source coupling. FIPS 186-5 characterizes digital signatures as providing evidence against unauthorized data modification and authenticating the signatory.[5] Those properties are important, but they do not by themselves establish that the signed observation remained coupled to the claimed physical source. If the acquisition boundary accepts a replayed or substituted stimulus, or becomes decoupled from the claimed physical source, the signer can faithfully protect bytes whose semantic premise is already invalid. Source-veracity evidence addresses the additional physical-source claim the application actually cares about.

The consequence is placement. If the evidentiary meaning of a signal depends on properties that exist only at capture, then those properties must be observed, captured, or bound before they become unrecoverable. The complete appraisal need not necessarily occur at the sensor. But evidence generated downstream cannot reconstruct a source property that the architecture allowed to disappear before binding. If freshness depends on protected time or monotonic state, that state must bind to acquisition where the claim remains meaningful. If an admission decision must be non-bypassable, it cannot exist solely as advisory metadata that downstream software is free to ignore.

> **Partitioning rule**
> The evidence determines the boundary. The boundary constrains the partition. The partition determines the substrate.

4. Signal Veracity Changes Hardware Architecture

Traditional hardware/software partitioning asks where a function runs most efficiently. Signal-veracity and observation-admission requirements add a second question: where must a property be observed, evidence be bound, or policy be enforced for the system claim to remain meaningful? This creates a placement pressure that can be called evidentiary locality.

4.1 Evidentiary locality

Evidentiary locality is the requirement that a claim-relevant property be observed, captured, generated, or cryptographically bound close enough to the relevant event, source, state, or protected boundary that the resulting evidence still supports the intended claim. It is not identical to low latency. A function can tolerate milliseconds of delay and still require local capture or binding because transporting an unbound raw observation first may permit replay, substitution, source decoupling, or loss of custody before the trust claim is formed.

Operationally, for each claim-relevant property p along a given acquisition-to-action path, define O_p as the earliest stage at which p can be observed and B_p as the last stage at which evidence for p can be bound without losing the intended meaning. A valid partition must capture or bind the required evidence no later than B_p and preserve its protected lineage thereafter.

The architecture therefore has to identify the earliest point at which each source-dependent property can be observed and the latest point at which it can still be bound without losing meaning. In some systems that point may be inside the sensor. In others it may be the first trusted FPGA, MCU, SoC, or acquisition gateway. The correct boundary is threat-model dependent. The principle is not that all veracity logic belongs in a sensor; it is that the trust claim should not depend on evidence generated after the property it is supposed to establish has become unrecoverable.

4.2 Protected admission before consequential downstream compute

The useful boundary is not 'before all computation,' because extracting and appraising evidence are themselves computations. The useful boundary is before consequential downstream computation or actuation treats the observation as admitted. A protected acquisition-and-admission path may therefore include acquisition, bounded preprocessing, signal-derived evidence generation, freshness and session binding, local appraisal, and admission enforcement. Alternatively, it may produce protected evidence for downstream appraisal, provided that source-dependent properties were bound before they were lost. The output is not merely data and should not be a single trust score; it is data plus structured evidence or an admission state that downstream policy can verify.

A downstream accelerator interface can therefore be understood conceptually as: observation + protected evidence reference + provenance + freshness + admission state + policy context -> eligible downstream computation. This is different from a conventional tensor-in, result-out contract. Trust becomes part of the compute interface rather than metadata attached after the fact.

4.3 Keep evolving evidence programmable; harden stable invariants

Signal science evolves. Thresholds move. Feature families change. Adversarial testing reveals failure modes. Modalities differ. A premature ASIC can freeze the least mature part of the trust model. The first hardware response should therefore not be 'put the veracity algorithm in fixed silicon.' It should be to separate stable trust invariants from evolving signal-derived evidence.

Stable invariants may include device identity, protected freshness state, key isolation, authenticated provenance, non-bypassable admission policy, quarantine state, revocation, secure update, and recovery. Platform firmware guidance already treats protection, detection, and recovery as distinct resiliency obligations.[6] Modality-specific evidence extraction, calibration, classifiers, thresholds, and adversarially responsive logic may need to remain software- or FPGA-programmable longer. Only kernels that survive empirical evaluation, implementation evidence, and lifecycle analysis should become candidates for hardening. This is the same irreversibility discipline developed in Analysis 001, now applied to trust.[7]

Hardware rooting is not a substitute for empirical validity. A protected bad detector is still a bad detector, and hardening an immature evidence model merely makes error more expensive to change. The architecture should therefore freeze invariants before it freezes algorithms. Programmability is not evidence of weak trust when it is deliberately contained inside a protected boundary; it can be the correct substrate for a trust function whose science is still evolving.

4.4 Trust changes memory and control planes

Once admission depends on evidence, memory can no longer be treated as one undifferentiated logical state. An architecture may need explicit ownership or labeling for unverified observations, admitted observations, quarantined observations, provenance-bound derived features, and provenance-bound results. That distinction does not require physically separate memories in every implementation; it requires enforceable state and access semantics. DMA ownership, access control, buffer reuse, and accelerator scheduling can depend on those semantics. The control plane must also define what the system does when evidence is insufficient: reacquire, quarantine, restrict inference, buffer for later review, operate in a validated degraded mode, or refuse a consequential action.

A further problem appears after feature extraction. If an admitted observation becomes a derived feature vector, tensor, event stream, or compressed representation, the architecture has to preserve lineage between that derived object and the evidence supporting the admitted observation. Otherwise provenance can disappear exactly at the hardware/software boundary where acceleration begins. A protected handle, authenticated metadata, or equivalent binding can allow downstream compute to verify that its input descends from an admitted observation without requiring every accelerator to re-evaluate the original signal.

A trustworthy system therefore requires a designed refusal path: no admission, no consequential compute or actuation, or a deliberately bounded degraded mode when required evidence is absent. The processor may still execute housekeeping or recovery logic; the point is that missing evidence must not be silently converted into assumed trust by operational pressure. Fail-closed is not universally safe: in medical, industrial-control, or autonomy settings, refusal can itself be consequential. Degraded-mode and override policy should therefore be validated against both false-accept and false-reject risk, with any override explicit, policy-bound, and provenance-preserving.

5. Caliptra Shows Why Placement Matters

Caliptra is an instructive industry anchor because it makes a placement argument without making a signal-veracity claim. The CHIPS Alliance project defines IP and firmware for an integrated Root of Trust block targeting datacenter-class SoCs and provides identity, measured boot, and attestation capabilities.[3] The Caliptra 2.0 specification defines core Silicon Root of Trust capabilities to be implemented in the SoC or ASIC, including roots for measurement and identity. The design therefore makes placement part of the trust claim: protected state, keys, measurements, and attestation are anchored where lower-trust software cannot simply redefine the root evidence.

That architecture illustrates a broader principle: when the validity of a trust claim depends on where evidence is generated and protected, trust becomes a partitioning constraint. Caliptra demonstrates the industry logic of integrating machine identity and measurement into a silicon root. Signal-veracity requirements raise the analogous architectural question farther upstream: which source-dependent properties must be captured or bound close enough to acquisition that their meaning survives transport and downstream computation?

The two claims are complementary, not competing. Machine evidence asks whether silicon identity, measured boot, and security-relevant state support appraisal under the applicable policy. Observation evidence asks whether the physical input presented to that machine satisfies the source-coupling and related freshness, provenance, and admission requirements of the application. A high-consequence result may require both before the relying party should act.

6. High-Consequence Distributed Systems

The partitioning effect becomes strongest when acquisition and consequential compute are separated by distance, intermittent connectivity, multiple administrative domains, or changes in custody. Medical sensing, defense and autonomy, industrial monitoring, and remote critical infrastructure all create versions of this problem. The important architectural question is not whether the cloud or data center is more powerful. It is which properties must already have been observed or bound before the observation leaves the place where those properties can still be established.

In such systems, a remote endpoint should not be the first component asked to establish source-coupling evidence when the relevant source property would be lost before transport and the threat model includes replay, substitution, source decoupling, or loss of custody. The edge may therefore need enough protected state and bounded compute to bind identity, freshness, provenance, and source-veracity evidence into an admission-relevant record even while disconnected. Heavier inference can still occur downstream. The point is not to move every model to the edge; it is to prevent remote compute from being asked to reconstruct a physical-source claim after the architecture has already lost the evidence needed to support it.

The same principle applies inside heterogeneous and multi-die systems. An accelerator or chiplet should not infer admissibility from the mere arrival of a packet or DMA transaction. If the application requires admitted input, the interface contract should carry or reference the relevant evidence, provenance, freshness, authorization, and admission state, and the receiving domain should have a defined behavior when that state is absent or invalid. This turns trust from a platform-side service into an architectural property of data movement.

This also changes failure architecture. A distributed system should distinguish at least these operational conditions: admitted under current policy; degraded but bounded; quarantined or evidence-insufficient; compromised or lost-custody; and recovery or re-enrollment. Re-entry should require explicit recovery, re-enrollment, or reacquisition rules rather than silently restoring the prior admission state after connectivity or custody returns.

Exhibit 2. Partitioning pressures introduced by evidentiary requirements

Architecture pressure	Likely partitioning push
High throughput / repeatability	Dedicated hardware or accelerator
Tight latency / deterministic timing	Edge or hardware execution
Rapidly changing evidence algorithm	Programmable software / FPGA
Source-coupling requirement	Near-acquisition evidence capture / binding
Custody / provenance requirement	Protected trust boundary
Non-bypassable admission policy	Hardware-rooted enforcement
Intermittent network	Local trusted state + bounded local compute
High consequence of false acceptance	Quarantine / refusal / reacquisition path
Mature persistent trust kernel	Candidate for hardened silicon
Derived-data lineage requirement	Provenance-bound features / tensors / event streams
Cross-domain accelerator or chiplet boundary	Evidence-aware interface + admission enforcement

7. A Signal-to-Substrate Partitioning Test

A practical architecture review can treat observation trust as a sequence of placement and enforcement questions rather than a generic security requirement. The objective is to identify the consequential claim, the physical properties that must remain true, where evidence for those properties can still be captured or bound, what policy must be non-bypassable, what should remain programmable, and what evidence would cause the selected partition to change.

	Question	Architecture consequence
1	What consequential claim or action is the system making?	Name the decision or action that becomes invalid if the observation is replayed, substituted, stale, source-decoupled, or processed on an unacceptable machine state.
2	Which source-dependent properties must remain true, and what companion evidence is required?	Separate source coupling from freshness, provenance, device/session binding, quality, and machine evidence rather than collapsing them into one trust score.
3	Where can each property still be observed or bound?	Locate the earliest observation point and the latest meaningful binding point. This defines evidentiary locality.
4	Which appraisal or enforcement functions must be non-bypassable?	Identify admission, quarantine, freshness state, key use, provenance sealing, and recovery rules that cannot be left to advisory software.
5	Which functions are still scientifically or operationally volatile?	Keep modality-specific evidence, thresholds, classifiers, calibration, and adversarially responsive logic programmable until evidence supports hardening.
6	What substrate matches the maturity and placement of the claim?	Use software when assumptions are volatile, FPGA/adaptive hardware when protected placement matters but logic is evolving, and fixed silicon only when the trust kernel is stable, valuable, and economically justified.
7	What evidence would cause the partition to change?	Define adversarial tests, failure thresholds, update cadence, and kill conditions before hardening so contradictory evidence can force repartition rather than be rationalized away.

This test does not require every system to adopt a new hardware block. It requires the architecture to state explicitly where observation evidence is created or bound, how it is protected, where it is appraised, what happens when it fails, and whether the selected substrate preserves the required claim throughout the system. The outcome may be sensor-resident evidence capture or binding, an FPGA-resident trust plane, a protected acquisition gateway, a hardware-enforced admission interface, or a deliberately software-defined evidence model inside a protected environment. The right answer depends on the claim, threat model, evidence maturity, and lifecycle economics.

For an architecture office, this means trust belongs in the same trade study as latency, bandwidth, power, memory movement, lifecycle, test, and cost. If changing the trust boundary changes where buffers live, which domain owns keys, whether disconnected operation is admissible, whether a chiplet can accept a transaction, or whether a function may be hardened, then trust has already become a first-order partition variable. Treating it as a post-architecture security review simply delays the decision until it is more expensive to change.

8. The Beginning of Trust

A silicon root of trust should not be confused with the beginning of the evidentiary chain. A silicon root of trust can anchor identity, measurement, protected state, and attestation for the computing platform. In a sensor-driven system whose consequential claim depends on physical state, however, the evidentiary chain begins earlier: at the observation and acquisition boundary where the physical premise can still be tested or bound.

The architecture therefore has to bind two evidentiary domains: observation evidence and machine evidence. Combined with policy, freshness, provenance, and protected execution, they can support a stronger end-to-end claim than either domain alone. The resulting design principle is simple: partition according to where the claim must remain true.

> **Architecture thesis**
> Performance determines where computation is efficient. Trust determines where computation remains valid.

For architects, this changes the substrate decision. Some functions justify hardware-rooted enforcement not because they dominate FLOPs, but because the system claim depends on placement, protected state, or non-bypassability. Other functions should remain programmable precisely because the evidence is not yet mature enough to harden. Trust is therefore not a security feature added after partitioning. In consequential systems, it is one of the forces that determines the partition itself.

The practical design sequence is therefore: define the consequential claim; identify which source-dependent properties must remain true; determine where evidence for each property can still be observed or bound; establish the appraisal,

admission, and failure behavior; separate stable invariants from evolving evidence logic; define what would invalidate the partition; and only then select the substrate. Signal -> evidence requirement -> trust boundary -> partition -> substrate. That sequence makes the trust claim testable before it becomes an irreversible hardware assumption.

Living Cipher's related bio-cryptographic attestation research treats live-source witness evidence as a distinct layer that can be cryptographically bound into a verifier-facing receipt.[8] The present paper makes a different contribution: it asks what an upstream source-evidence requirement does to the system partition, data path, control plane, interface contract, and eventual silicon boundary. The signal-veracity mechanism can evolve; the architecture still has to decide where the required claim becomes protected and non-bypassable.

Scope and disclosure boundary.

Living Cipher develops mechanisms for deriving signal-veracity evidence and binding that evidence into hardware-rooted trust decisions. This paper presents the public architectural thesis and deliberately separates that thesis from proprietary implementation machinery. It does not disclose private witness construction, scoring or normalization methods, thresholds, dependence modeling, tamper-decision logic, or implementation topology.

Declaration of Generative AI and AI-Assisted Technologies in the Writing Process

During the preparation of this work, the author used ChatGPT (OpenAI) primarily to assist with manuscript organization, structural editing, and refinement of language and presentation. The architectural and analytical framework, technical analysis, interpretation of evidence, judgments, and conclusions are the author's own. The author reviewed, verified, and edited the manuscript and takes full responsibility for its content.

Selected References

1. N. Panwar, S. Sharma, G. Wang, S. Mehrotra, N. Venkatasubramanian, M. H. Diallo, and A. Amiri Sani, “IoT Notary: Attestable Sensor Data Capture in IoT Environments,” ACM Transactions on Internet of Things, 3(1), 2022. DOI: 10.1145/3478290.
2. H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, “Remote ATtestation procedureS (RATS) Architecture,” IETF RFC 9334, January 2023. DOI: 10.17487/RFC9334. https://www.rfc-editor.org/rfc/rfc9334.html
3. Open Compute Project / CHIPS Alliance, “Caliptra: A Datacenter System on a Chip (SoC) Root of Trust (RoT),” Version 2.0, July 2025. https://github.com/chipsalliance/Caliptra/blob/main/doc/caliptra_20/Caliptra.ocp
4. DMTF, DSP0274, “Security Protocol and Data Model (SPDM) Specification,” Version 1.4.0, May 25, 2025. https://www.dmtf.org/dsp/DSP0274
5. National Institute of Standards and Technology, FIPS 186-5, “Digital Signature Standard (DSS),” February 2023. DOI: 10.6028/NIST.FIPS.186-5. https://csrc.nist.gov/pubs/fips/186-5/final
6. National Institute of Standards and Technology, SP 800-193, “Platform Firmware Resiliency Guidelines,” May 2018. DOI: 10.6028/NIST.SP.800-193. https://csrc.nist.gov/pubs/sp/800/193/final
7. Dana Xiadani, “From Signal to Substrate: Architecture, Irreversibility, and the Capital Economics of Custom Silicon,” Living Cipher Analysis 001, Version 1.0, 2026. DOI: 10.5281/zenodo.21958390.
8. Dana Xiadani, “Towards Bio-Cryptographic Attestation: A Candidate Framework for Live-Source Witnessing and Receipt-Bound Evidence,” Living Cipher, working paper, 2026.

Author Note

Dana Xiadani is Founder & Chief Architect of Living Cipher, where she works at the boundary of signal, computation, architecture, trust, and physical realization. Living Cipher Analysis examines decisions upstream of implementation and capital commitment: what computation deserves to become hardware, which assumptions are being capitalized, where technical choices become dependencies, and what evidence should exist before a system becomes expensive to change. This paper is an architecture analysis; it does not claim empirical validation of any particular source-veracity detector or modality.

Suggested citation: Xiadani, Dana. 2026. “The Trusted Chip Is Not the Trusted Signal: How Signal Veracity Changes Hardware Partitioning in Consequential Systems.” Living Cipher Analysis 002, Version 1.0. DOI: 10.5281/zenodo.22019132.