A Framework for

Continuous Remote Attestation

in Confidential Computing Environments

Authors: *The MITRE Corporation* Dr. Mari Spina, Gary Warner, Dr. Paul Rowe, Dr. Joshua Guttman, Andy Radle, Paul Vanderveen

Fr0ntierX Inc.: Jens Albers, Jonathan Begg, Andrew Gilbert, Nick Renner

Invary Inc.: Jason Rogers, Dr. Perry Alexander

Version 0.9.6 – Public Review Draft

May 2026

Sponsor:

Dept. No: L523

Contract No:

Project No:

McLean, VA

Prepared by:

©2026 The MITRE Corporation. All rights reserved.

Approved for public release. Distribution unlimited 26-00364-1

Notices

The views, opinions, and findings contained in this document are those of The MITRE Corporation and contributing organizations and should not be construed as an official government position, policy, or decision, unless designated by other documentation.

This document has been developed through a collaborative working group comprising MITRE, Fr0ntierX Inc., and Invary Inc. Intellectual property contributed by each party remains the property of the contributing organization. Jointly developed concepts within this framework are jointly owned and may be freely referenced and implemented.

This document defines an open architectural framework. It does not endorse specific commercial products or implementations. References to specific technologies or products are included solely for technical illustration.

Acknowledgments

This framework was developed under the Confidential Computing Layered Attestation Working Group.

Document Status

This is a Public Review Draft (v0.9.5) published to solicit written feedback from the confidential computing community. Comments should be directed to the working group through the designated feedback channels. A final specification will be published following the review period, incorporating community input. The working group will subsequently invite broader collaboration during the open-source reference implementation phase.

Revision History

Version	Date	Description
0.9.6	May 2026	Public Review Draft. Diagrams, Threat Model Revision.
0.9.5	May 2026	Public Review Draft. Trust Anchor Composition (Section 4.1.3), Provider Trust Stance (Section 5.3), Annex B Provider Attestation Service Architecture, perspective-table stance defaults.
0.9.4	April 2026	Public Review Draft. Post-Quantum Requirements
0.9.3	April 2026	Public Review Draft. Added ESTM Technique Mapping

Version	Date	Description
0.9.2	April 2026	Public Review Draft. SPIFFE / SPIRE, Consistency, minor edits
0.9.1	April 2026	Public Review Draft. Added Alternatives to Yang, formatting edits, Edits according to ICS, On-Demand Attestation
0.9.0	March 2026	Public Review Draft. Added AI workload attestation controls, GPU TEE attestation guidance, multi-perspective consumer model, attestation interval policy, transport binding deployment constraints, continuous status visibility, compliance control mapping annex.
0.2.0	February 2026	Internal working draft. Incorporated Split Verifier architecture, YANG Push publisher protocol, threat model (EMB3D + Cloud Overlay).
0.1.0	January 2026	Initial threat model and four-layer evidence model.

Table of Contents

List of Figures

List of Tables

1. Executive Summary

Current approaches to remote attestation in cloud computing environments are overwhelmingly static: they verify a workload's integrity at boot time and assume that initial state persists throughout the workload's lifetime. This assumption is fundamentally incompatible with modern Zero Trust architectures, where trust must be continuously earned and never assumed.

This document defines a standardized, open architecture for Continuous Remote Attestation of workloads running inside Trusted Execution Environments (TEEs) across cloud, edge, and on-premise deployments. The framework is designed to be implementation-neutral and cloud-agnostic, establishing the roles, evidence layers, protocols, and consumer perspectives necessary for interoperable continuous attestation.

The architecture extends the Internet Engineering Task Force (IETF) Remote ATtestation procedureS (RATS) model defined in Request for Comments (RFC) 9334 with four key contributions:

- **A three-layer continuous evidence model** (Platform, Image, Runtime) that moves attestation from a single boot-time check to ongoing verification of what is running and how it behaves.

- **A multi-perspective consumer model** that accounts for the distinct trust requirements of Cloud Service Providers, Infrastructure Consumers, Application Developers/Operators, and End Users.

- **AI workload attestation controls** that address the unique integrity challenges of machine learning inference and training pipelines operating within confidential computing environments.

- **Graphics Processing Unit (GPU) TEE attestation guidance** that extends the platform validation layer to incorporate accelerator-based TEEs for AI training and inference workloads.

The framework does not prescribe specific implementations. Instead, it defines the architectural roles, evidence schemas, protocol requirements, and security properties that conforming implementations must satisfy. This separation of specification from implementation enables competitive innovation while ensuring interoperability.

2. Introduction and Motivation

2.1 The Attestation Gap

Confidential computing technologies – Advanced Micro Devices (AMD) Secure Encrypted Virtualization-Secure Nested Paging (SEV-SNP), Intel Trust Domain Extensions (Intel) (TDX), Arm Confidential Compute Archive (CCA), NVIDIA Confidential Computing (CC) – provide hardware-enforced memory isolation that protects workloads from privileged software, including the hypervisor and host operating system. These technologies generate cryptographic attestation reports signed by hardware-fused keys, enabling remote parties to verify that a workload is running inside a genuine TEE.

However, most current attestation practices suffer from a critical gap: they verify the state of the TEE at a single point in time (typically boot) but provide no continuous assurance that the running workload has not been compromised after initial measurement. A workload that passes boot-time attestation may subsequently load unauthorized code, suffer a kernel rootkit, or begin exfiltrating data – none of which would be detected by a static attestation model.

This framework closes that gap by defining a layered, continuous attestation architecture where evidence is gathered and appraised at regular intervals throughout the workload's lifetime, and trust decisions are updated in near real-time.

2.2 Scope and Audience

This framework is intended for security architects, platform engineers, compliance officers, and standards bodies working on confidential computing deployments. The framework also applies to Industrial Control System (ICS) and Operational Technology (OT) environments where TEE-based isolation is deployed at Information Technology (IT)/OT convergence points – including protocol-translation gateways between enterprise and control networks, historian databases aggregating process data, and edge controllers executing supervisory logic. In these environments, continuous attestation provides assurance that the software mediating access between IT and OT domains has not been modified or compromised, a property that is critical as IT/OT convergence expands the attack surface of previously air-gapped industrial systems.

It is applicable to cloud (IaaS, PaaS, SaaS), edge, and tactical / on-premise environments where TEE-based workload isolation is employed.

The document is structured as follows: Sections 3-4 define the architectural roles and evidence layers. Section 5 introduces the multi-perspective consumer model. Section 6 addresses AI-specific attestation controls. Section 7 defines the publisher protocol, attestation interval policy, and anti-replay mechanisms. Section 8 defines attested transport binding requirements and deployment constraints. Section 9 presents the

threat model. Section 10 discusses standards alignment. Section 11 provides implementation guidance. Annexes provide the compliance control mapping methodology.

2.3 Relationship to Existing Standards

This framework builds upon and extends the following standards:

- **IETF RFC 9334 (RATS Architecture):** This framework adopts the RATS roles (Attester, Verifier, Relying Party, Endorser, Reference Value Provider) and extends them with the Split Verifier model required for continuous attestation of confidential workloads.

- **IETF EAT (Entity Attestation Token):** Evidence format for platform attestation tokens, with eat_nonce binding for freshness.

- **MITRE Embedded Threat Model framework (EMB3D):** Informs the hardware and firmware threats (Layer 1), with a Cloud Overlay for hypervisor and orchestration threats.

- **National Institute of Standards and Technology (NIST) AI Risk Management Framework (RMF) 1.0 / Cyber AI Profile (NIST Interagency Report (IR) 8596):** Informs the AI workload attestation controls in Section 6 of this framework.

- **Secure Production Identify Framework for Everyone (SPIFFE) / SPIFFE Runtime Environment (SPIRE):** SPIFFE defines interoperable workload identities and credential formats for service-to-service authentication. SPIRE is a reference implementation that automates workload identity issuance, rotation, and revocation. Within this framework, SPIFFE / SPIRE is a complementary identity layer that may consume attestation results as a policy input. It does not replace hardware-backed attestation, continuous appraisal, or Attested Transport Binding. Implementations may gate issuance and renewal of SPIFFE Verifiable Identity Documents (SVIDs) on the current attestation verdict so that only Target Machines in a PASS state receive workload credentials.

NOTE: SPIRE's native workload attestation mechanisms (node attestation, workload registration selectors) evaluate platform metadata and scheduling state. These mechanisms are distinct from the hardware-rooted, continuous integrity evidence defined by this framework's three-layer evidence model and do not satisfy the framework's attestation requirements.

3. Architectural Roles (Split Verifier Model)

The framework defines six architectural roles. A given physical or logical component may fulfill one or more roles simultaneously. This role decomposition extends the IETF RATS model (RFC 9334) with the Split Verifier pattern required for continuous in-boundary attestation.

Table 1. Architectural Roles (Split Verifier Model)

Role	Function	Input	Output
Target Machine	The TEE-protected workload environment. Contains the hardware Root of Trust and runs the customer workload.	N/A (source of evidence)	Raw hardware attestation reports, runtime telemetry
Measurement Data Gatherer	Collects raw evidence from the Target Machine at each attestation layer (L1, L2, L3). Operates inside the TEE boundary.	Kernel measurements, Integrity Measurement Architecture (IMA) logs, Extended Berkeley Packet Filter (eBPF) events, container digests	Evidence Bundle (structured, signed)
Reference Data Gatherer	Retrieves expected reference values (Golden Baselines) from trusted supply chain sources.	Vendor signatures, Software Bill of Materials (SBOMs), CoRIM manifests, policy definitions	Composite Reference Baseline
Appraiser	Compares measured evidence against reference baselines. Produces a trust verdict per layer and an overall appraisal.	Evidence Bundle + Composite Reference Baseline	Appraisal Result (PASS / ALERT / FAIL per layer)
Publisher	Streams appraisal results to Enterprise Validators. Responsible for anti-replay, freshness, and signed delivery.	Appraisal Results	Signed events via Yet Another Next Generation (YANG) Push Server-Sent Events (SSE), REST Application Programming Interface (API), or webhooks
Enterprise Validator	The organization’s relying party. Consumes published attestation results and enforces trust policy, such as revoking Key Management Service (KMS) access, blocking traffic, raising SIEM alerts, or driving workload identity decisions via external identity systems.	Published Appraisal Results	Trust enforcement actions

The Enterprise Validator may integrate with an external workload identity system, such as SPIFFE / SPIRE, to translate attestation verdicts into workload credential issuance, renewal, or revocation decisions. In this integration model, the attestation framework remains the trust source, and the workload identity system operationalizes that trust state for service-to-service authentication and authorization.

3.1 The Split Verifier Pattern

A critical architectural decision in this framework is the separation of the Measurement Gatherer from the Appraiser, with the Appraiser operating inside (or adjacent to) the TEE boundary rather than externally. “Adjacent to” includes architectures where the Appraiser runs as a co-located sidecar container sharing the same TEE boundary, or as a service at a higher privilege level within the same hardware isolation domain (e.g., a Secure Virtual Machine (VM) Service Module at Virtual Machine Privilege Level 0 (VMPL0) providing appraisal services to a guest at VMPL2 or VMPL3). This “Split Verifier” pattern provides three key benefits:

- **Privacy:** Raw evidence (kernel memory layouts, binary hashes, process trees) never leaves the trust boundary. Only the signed appraisal result is published externally.
- **Bandwidth:** Kernel integrity validation may examine gigabytes of memory structures. Exporting this raw data would be impractical at continuous attestation intervals.
- **Integrity:** The appraisal itself is conducted in a protected environment, reducing the attack surface for result manipulation.

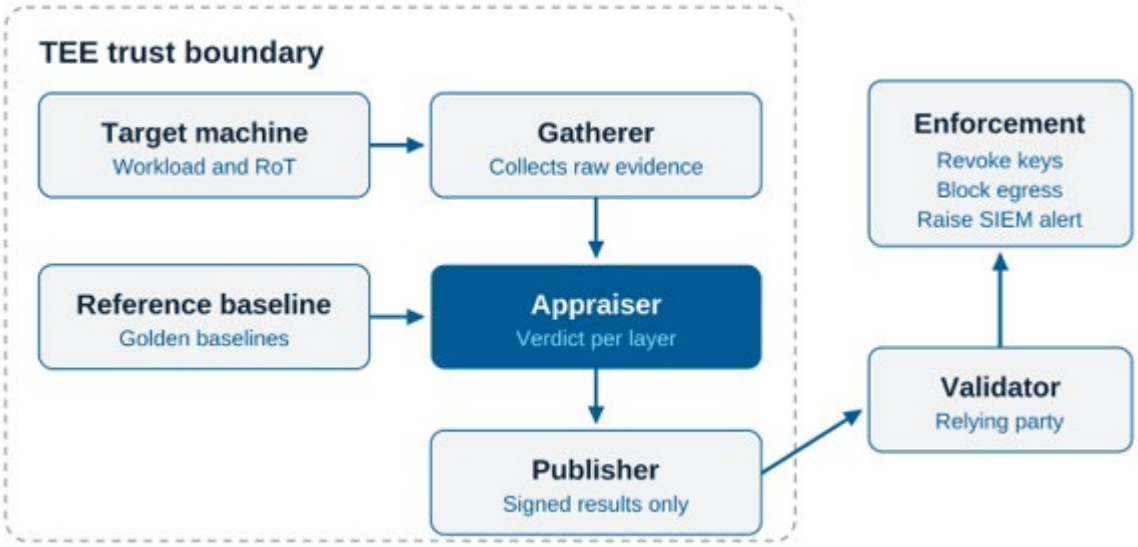

Figure 1. Split Verifier architecture and appraisal result flow

4. Three-Layer Evidence Model

The framework defines three layers of attestation evidence, each addressing a distinct trust question. Layers are evaluated independently and composed into an overall trust verdict. This layered approach enables implementations to support incremental deployment – an operator may begin with Layer 1 only and progressively enable deeper layers as their security requirements demand.

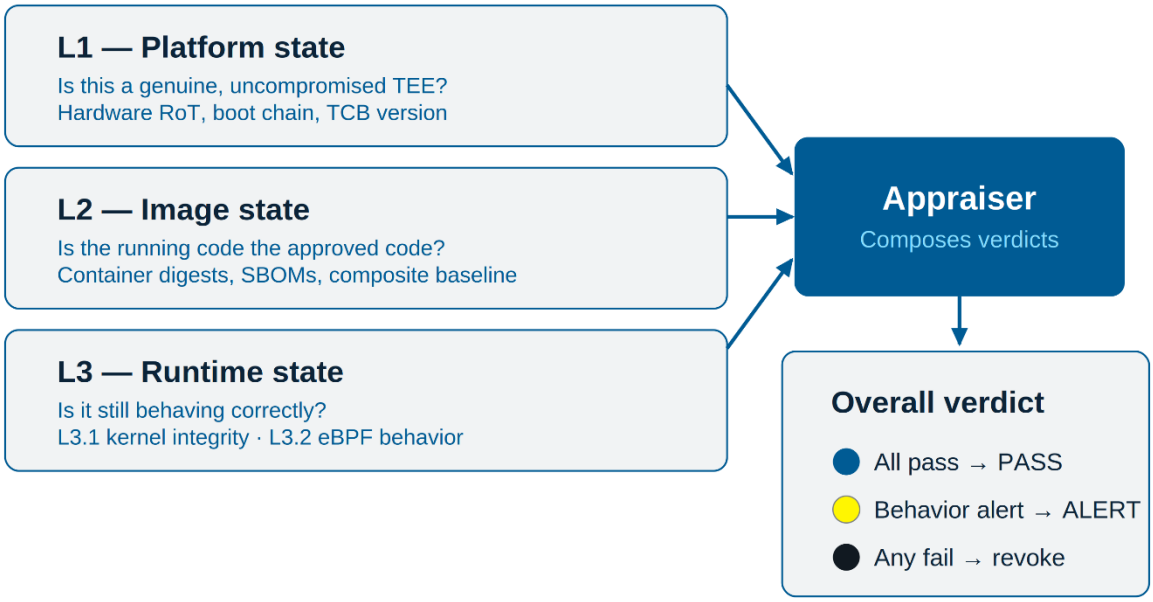

Figure 2. Three-layer evidence model and verdict composition

The "integrity dial" metaphor: Runtime attestation is not binary. The framework establishes the baseline requirements at each layer, while implementations choose how far to advance based on use case, performance constraints, and risk tolerance.

4.1 Layer 1: Platform State (TEE Identity and Firmware)

Trust Question: "Is this workload running inside a genuine, uncompromised TEE with known-good firmware?"

Layer 1 validates the hardware identity of the TEE and the integrity of the firmware and boot chain. Evidence originates from the hardware Root of Trust (e.g., AMD Platform Security Processor (PSP), Intel Software Guard Extensions/Trust Domain Extensions (SGX/TDX), Arm Confidential Compute Architecture (CCA)) and is typically conveyed as a signed attestation report or Trusted Platform Module (TPM)-based JSON Web Token (JWT) token.

4.1.1 L1 Evidence Requirements

Table 2. L1 Platform Evidence Requirements

Evidence Element	Description	Freshness Mechanism
Hardware Attestation Report	Signed by the platform's hardware Root of Trust (e.g., AMD Versioned Chip Endorsement Key (VCEK), Intel Quoting Enclave (QE)). Contains launch measurements, Trusted Computing Base (TCB) version, debug status.	eat_nonce binding or challenge-response nonce in platform-specific user-data field
Boot Chain Measurements	PCR values or equivalent recording the firmware, bootloader, kernel, and initrd hashes measured during Secure Boot.	Included in attestation report
TCB Version	Minimum version assertion for boot_loader, TEE, SNP/TDX microcode. Rejects rollback to vulnerable firmware.	Policy-enforced minimum (min_tcb_version)
Debug Status	Assertion that debug interfaces are disabled since boot.	Claim in attestation report (dbgstat)
Transport Binding	Cryptographic binding between the attestation report and the Transport Layer Security (TLS) session key, as defined by the Attested Transport Binding requirements (TB-1 through TB-6) in Section 8.	Per-session freshness (TB-2)

L1 Verdict: PASS or FAIL. Failure triggers immediate trust revocation. There is no ALERT state for platform integrity.

NOTE: Transport Binding appears as an L1 evidence element because the attestation report that proves platform authenticity is the same artifact used to satisfy the TB-1 key commitment (Section 8). The evaluation order is sequential, not circular: the Appraiser first validates L1 platform evidence (hardware attestation report, firmware versions, Secure Boot status), and the client then verifies transport binding as a post-condition by confirming that the validated attestation report contains a commitment to the TLS session's public key. L1 attestation establishes what the platform is; transport binding establishes that the client is talking to it.

4.1.2 L1-GPU: Accelerator TEE Attestation (Informative)

AI training and inference workloads increasingly execute within GPU-based TEE (e.g., NVIDIA CC on H100/H200 GPUs). When a Target Machine includes a GPU TEE, the platform validation layer extends with an L1-GPU sub-layer that validates the GPU hardware independently of the Central Processing Unit (CPU) TEE.

L1-GPU attestation operates as a companion to L1 CPU attestation, not a replacement. Both validations are required when GPU TEE attestation is enabled by the deployment baseline. The GPU attestation report is signed by the GPU vendor's hardware Root of Trust, which is independent of the CPU vendor's Root of Trust.

Table 3: L1-GPU Accelerator Evidence Requirements

Evidence Element	Description	Verification Method
GPU Hardware Attestation Report	Signed by the GPU vendor's Root of Trust (e.g., NVIDIA NRAS EAT token). Contains GPU model, firmware version, confidential computing mode status, driver version.	Signature verification against GPU vendor JWKS or certificate chain
GPU Firmware / Driver Version	Minimum version assertion for GPU driver and VBIOS firmware. Rejects rollback to vulnerable GPU firmware.	Policy-enforced minimum in baseline
GPU Confidential Computing Mode	Assertion that the GPU is operating in confidential computing mode with memory encryption enabled.	Claim in GPU attestation report
Shared Nonce Binding	The GPU attestation nonce MUST be a deterministic function of the same inputs used for the CPU attestation nonce (e.g., the TLS public key hash and the Appraiser's challenge nonce), such that the Appraiser can independently re-derive and verify the GPU nonce without additional out-of-band information. This cryptographically binds both CPU and GPU attestation to the same TLS session and prevents cross-device replay.	Nonce re-derivation and comparison by the Appraiser

L1-GPU Verdict: PASS, FAIL, or N/A. When the deployment baseline requires GPU attestation (gpu_required: true), L1-GPU FAIL triggers immediate trust revocation. When GPU attestation is optional and no GPU is present, L1-GPU returns N/A and does not affect the overall verdict.

NOTE: This section is informative. GPU TEE attestation architectures are evolving and vary by vendor. The framework defines the abstract requirements (shared nonce binding, independent verification, baseline-controlled enforcement) without mandating a specific GPU attestation protocol. Implementations should document their GPU attestation model, the GPU vendor's Root of Trust, and any trusted third-party dependencies introduced by the GPU attestation path.

Deployments with multiple GPU TEEs (e.g., 8×H100 configurations common in AI inference and training) must produce per-GPU attestation reports. A FAIL on any individual GPU constitutes an L1-GPU FAIL for the Target Machine. The attestation metadata must include the total GPU count and per-GPU verdict breakdown. The scalability implications of per-GPU attestation at short reporting intervals are deployment-specific and should be evaluated during capacity planning.

4.1.3 Trust Anchor Composition

Hardware attestation evidence reaches the relying party through a certificate chain or a signed token path. The composition of that path determines what the relying party is

implicitly trusting beyond the silicon itself. Different deployments and different consumer perspectives have legitimately different requirements for what entities they accept in this path. The framework defines three trust anchor types so that implementations can disclose composition and relying parties can make informed acceptance decisions.

Type A: Silicon-Rooted. The certificate chain terminates at the hardware vendor's public root with no operating service interposed between the silicon and the relying party. Examples include AMD ARK chains terminating at VCEK, Intel SGX Root Certificate Authority (CA) chains terminating at Provisioning Certification Key (PCK), Arm CCA platform roots, and IBM Universal Public Key chains for Secure Execution on Z. The relying party trusts the silicon vendor's Public Key Infrastructure (PKI) and nothing else in the verification path.

Type B: Provider-Mediated. The chain includes a Cloud Service Provider credential or a CSP-operated verification service between the silicon root and the relying party. The CSP performs a portion of the verification on the relying party's behalf and typically returns a signed token asserting the underlying hardware report's validity. The relying party trusts the silicon vendor PKI and additionally trusts the CSP as an operating service in the verification path.

Type C: Vendor-Service-Mediated. The chain or token is issued by an operating service run by the hardware vendor itself, distinct from the silicon PKI. The relying party trusts the hardware vendor as a service operator in addition to as a silicon manufacturer.

A Target Machine may mix types across evidence elements. For example, a confidential AI deployment may present a Type A trust anchor for the CPU TEE and a Type C trust anchor for the GPU TEE. The trust anchor type for each evidence element shall be disclosed in the attestation metadata so that relying parties can apply their own acceptance policy. Trust anchor disclosure is a property of the evidence itself and applies regardless of whether the evidence is delivered through Attested Transport Binding (Section 8) or through the Publisher protocol (Section 7).

Trust anchor types are descriptive, not hierarchical. Type A is not categorically superior to Type B or Type C in the abstract. Each type imposes a different trust burden on the relying party, and the appropriate type depends on the consumer perspective (Section 5) and the deployment context. Different consumers will legitimately accept different combinations of trust anchor types as a matter of policy. The framework names this policy choice the Provider Trust Stance, defined in Section 5.3.

Implementer guidance: A relying party that accepts the Cloud Service Provider as part of its TCB can consume Type B evidence directly, and the framework's protocols compose with provider-operated verification services. A relying party that wants to trust only the silicon vendor consumes Type A evidence through hardware-direct paths or customer-operated verification. Both paths satisfy the framework's evidence requirements; the choice is a policy decision tied to a threat model. This decision is independent of the transport binding requirements in Section 8: a Type B evidence path

can be transport-bound, and a Type A evidence path may be delivered without transport binding when consumed via the Publisher protocol.

4.2 Layer 2: Image State (Static Application Identity)

Trust Question: “Is the software loaded inside the TEE exactly what was intended, and can its provenance be verified?”

Layer 2 validates the static identity of all software artifacts defined in the deployment’s baseline configuration that are loaded inside the TEE. This includes the operating system kernel, container images, sidecar components, and AI model artifacts. “All” refers to the artifacts enumerated in the Composite Reference Baseline (Section 4.2.2): the framework validates that every artifact declared in the baseline is present and unmodified, and that no undeclared executable artifacts are loaded (enforced via IMA). The baseline is curated by the operator and the platform vendor according to the Composite Baseline model; the framework does not require an exhaustive inventory of every byte on the system, but it does require that the set of measured artifacts is explicitly defined and that deviations from that set are detectable. Evidence is gathered at boot and at container lifecycle events.

4.2.1 L2 Evidence Requirements

Table 4: L2 Image Evidence Requirements

Evidence Element	Description	Verification Method
Container Image Digests	SHA-256 digest of each container image manifest running in the TEE.	Compared against Golden Baseline allowlist
Cryptographic Signatures	Sigstore/Cosign or equivalent signature over container images, verified against the trusted CI/CD signing key.	Signature verification against Vendor and/or Tenant public keys
IMA Boot Measurements	Linux IMA log capturing all executables, shared libraries, kernel modules, and firmware loaded at boot.	Hash comparison against known-good values in baseline
Software Bill of Materials (SBOM)	Signed SBOM (SPDX or CycloneDX) for each deployed artifact, enabling third-party authenticity verification.	Signature verification + dependency audit
Composite Baseline Override	Vendor-defined immutable reference values for platform components (kernel, runtime, monitoring agents). Tenant baselines cannot override these.	Two-tier comparison: vendor baseline takes precedence

L2 Verdict: PASS or FAIL. Any mismatch against the composite baseline constitutes a failure.
In the context of SBOM verification, "dependency audit" means verifying that every component declared in the SBOM is present at the declared version, that no undeclared components are loaded (by cross-referencing with the IMA boot log), and optionally

checking declared components against a vulnerability feed (e.g., Open Source Vulnerability database (OSV), National Vulnerability Database (NVD)). The framework requires the first two checks; vulnerability feed integration is deployment-specific and informative.

4.2.2 The Composite Baseline Defense

A distinguishing feature of this framework is the two-tier composite baseline model. To defend against the Malicious Tenant Insider threat – where a user with valid deployment credentials injects compromised software – the framework mandates that platform-level reference values (kernel, Operating System (OS), monitoring agents) are defined by the platform vendor and cannot be overridden by tenant-supplied baselines. This ensures that even a compromised tenant administrator cannot authorize a modified kernel or tampered monitoring agent.

The Appraiser must employ a discrimination mechanism – such as distinct signing keys, structurally separated namespaces, or an equivalent cryptographic provenance chain – that prevents a tenant with baseline configuration privileges from injecting, modifying, or overriding vendor-supplied reference values. The discrimination mechanism must be documented as part of the implementation's security architecture.

4.3 Layer 3: Runtime State (Dynamic Execution Behavior)

Trust Question: “Is the running workload behaving as expected, and has its execution environment remained uncompromised since boot?”

Layer 3 is the continuous attestation layer. Unlike Layers 1 and 2, which primarily validate state at boot or deployment events, Layer 3 evidence is gathered at regular intervals throughout the workload’s lifetime. The attestation interval is a policy parameter (see Section 7.4); the framework recommends a default of 60 seconds, configurable per deployment.
Layer 3 is subdivided into two domains reflecting distinct verification techniques and threat coverage:

4.3.1 L3.1: Kernel Space Integrity

Kernel integrity monitoring validates that critical kernel data structures have not been modified after boot. This addresses threats such as syscall table hooking, Interrupt Descriptor Table (IDT) manipulation, and Direct Kernel Object Manipulation (DKOM) rootkits.

Table 5: L3.1 Kernel-Space Integrity Evidence

Evidence Element	Description	Coverage
Syscall Table Hash	SHA-256 hash of the entire syscall table, compared against the golden value captured at boot.	Syscall hooking, function pointer modification
IDT Hash	SHA-256 hash of the Interrupt Descriptor Table entries.	IDT manipulation
Loaded Module List	Enumeration of all loaded kernel modules, compared against an explicit allowlist.	Unauthorized module insertion
Kernel Memory Invariants	Deep semantic verification of kernel memory layouts and pointer structures (requires specialized runtime integrity service).	DKOM, advanced rootkits, memory corruption

Implementation Note: L3.1 evidence gathering techniques range from eBPF-based kernel structure readers (basic coverage) to full kernel memory semantic verification services (comprehensive coverage). The framework does not mandate a specific technique but requires that the chosen method and its coverage limitations be disclosed in the attestation metadata.
Attestation metadata, as used throughout this framework, refers to structured information published alongside attestation evidence and appraisal results. It includes the active layer set, attestation interval, evidence gathering technique and its coverage limitations, deployment profile, and any other parameters necessary for an Enterprise Validator or End User to interpret the attestation results correctly. This concept aligns with the metadata fields described in IETF RFC 9334 (RATS Architecture) and is extended in this framework to include continuous attestation-specific parameters such as epoch identifiers, reporting intervals, and coverage gap disclosures.

Coverage Limitation: The evidence elements defined above detect modifications to static kernel data structures but do not detect dynamic function hooking via kprobes, ftrace handlers, or tracepoint callbacks. An attacker with sufficient privilege inside the TEE can attach hooks to arbitrary kernel functions – redirecting execution without altering the syscall table, IDT, or loaded module list. Detecting such hooks requires enumerating the active kprobe and ftrace registration lists and comparing them against an expected baseline. Implementations providing comprehensive L3.1 coverage should include active hook enumeration, implementations that do not must disclose this gap in their attestation metadata.

4.3.2 L3.2: Application and Process Space

Application-level runtime monitoring uses eBPF behavioral sensors and IMA runtime measurements to verify that the workload's execution matches its expected behavioral profile.

Table 6: L3.2 Application and Process-Space Evidence

Evidence Element	Description	Coverage
IMA Runtime Measurements	Incremental IMA log entries for executables loaded after boot.	Unauthorized code execution, library injection
Process Execution Events	All process creation (execve) events, matched against a behavioral allowlist.	Unexpected shell spawning, post-exploit activity
Network Connection Events	Outbound Transmission Control Protocol/User Datagram Protocol (TCP/UDP) connection attempts, compared against an egress allowlist.	Data exfiltration, C2 communication
Syscall Profile	Aggregated syscall frequency and type profile per process, compared against learned baseline.	Behavioral drift, anomalous activity
Domain Name System (DNS) Query Content (Optional)	DNS query names and response metadata from outbound DNS traffic, compared against an allowed domain list or anomaly baseline.	DNS tunneling, covert data exfiltration via query encoding

L3 Verdict: PASS, ALERT, or FAIL. The ALERT state allows behavioral monitoring in "learn mode" where deviations are logged but do not trigger trust revocation, enabling operators to refine their profiles before enforcement.

Limitations:

- L3.2 behavioral monitoring detects deviations from a learned or defined profile. Sophisticated adversaries who operate within the bounds of the behavioral profile will not trigger L3.2 violations. L3.2 is a defense-in-depth measure, not a guarantee of detection for all attack classes.

- L3.2 behavioral monitoring operates at the process and connection level. It detects deviations from a defined behavioral profile but does not inspect application-layer payload content. Covert channels that operate within permitted connections – including but not limited to DNS tunneling (data encoding in query labels over UDP:53), steganographic encoding in legitimate API traffic, and timing-based channels – are not detectable by L3.2 evidence alone. Deployments with high data sensitivity requirements should consider supplementary application-layer monitoring (e.g., DNS query content inspection, payload entropy analysis) as deployment-specific extensions to the evidence model. Such extensions, if implemented, should be reported in the attestation metadata under L3.2.

4.4 Decision Matrix

The overall trust verdict is derived from the individual layer verdicts according to the following composition rules.

Table 7: Verdict Decision Matrix

L1 Platform	L1-GPU	L2 Image	L3.1 Kernel	L3.2 App	Overall	Action
PASS	PASS or N/A	PASS	PASS	PASS	PASS	Maintain trust grant
PASS	PASS or N/A	PASS	PASS	ALERT	ALERT	Maintain grant; publish alert to Validator
FAIL	Any	Any	Any	Any	FAIL	Immediate trust revocation
Any	FAIL	Any	Any	Any	FAIL	Immediate trust revocation (GPU compromise)
Any	Any	FAIL	Any	Any	FAIL	Immediate trust revocation
PASS	PASS or N/A	PASS	FAIL	Any	FAIL	Trust revocation (kernel compromise)
PASS	PASS or N/A	PASS	PASS	FAIL	FAIL	Trust revocation (behavioral violation)

When the Target Machine includes additional hardware TEEs (e.g., GPU TEEs for AI workloads), the Appraiser must extend the decision matrix with an additional L1 sub-verdict per hardware TEE. A FAIL on any hardware TEE sub-verdict constitutes an overall FAIL. The L1-GPU guidance in Section 4.1.2 defines the evidence requirements for GPU TEE attestation.

For any combination of layer verdicts not explicitly listed above: any ALERT without a corresponding FAIL at any layer yields an overall ALERT. This composition rule applies regardless of which layer produces the ALERT.

Layers not active in the deployment profile (see Section 11.1) are treated as N/A and do not affect the overall verdict. The set of active layers must be disclosed in the attestation metadata. For example, a deployment using the Workload profile (L1 + L2) treats L3.1 and L3.2 as N/A.

The decision matrix defines the composition logic for combining layer verdicts into an overall trust state. The individual layer verdicts themselves are determined by layer-specific policies (reference baselines, behavioral allowlists, firmware version requirements) which are set by the stakeholders defined in the Multi-Perspective Consumer Model (Section 5). In general: L1 platform policy is informed by the silicon vendor and the CSP, who define acceptable firmware versions, TCB minimums, and hardware configuration requirements. L2 image policy is jointly governed by the platform

vendor (who supplies immutable vendor baselines for kernel and OS components) and the Application Operator (who defines tenant baselines for application containers), as described in the Composite Baseline model (Section 4.2.2). L3 runtime policy, including behavioral allowlists and kernel integrity check parameters, is set by the Application Operator based on the workload's expected behavior. The Appraiser evaluates measured evidence against all applicable policies and produces the per-layer verdict. The composition of those verdicts into the overall trust state follows the fixed rules in the matrix above and is not subject to policy variation.

5. Multi-Perspective Consumer Model

A critical insight from the working group is that attestation results are not consumed uniformly. Different stakeholders in a cloud computing stack have distinct trust requirements, visibility boundaries, and decision authorities. A single attestation output that conflates all perspectives fails to serve any of them well.

This section defines four attestation consumer perspectives, each with its own trust questions, evidence visibility, and actionable outcomes. The model draws from the nested deployment boundaries inherent in cloud computing: physical hardware contains TEEs, which contain virtual machines, which contain containers and application processes.

5.1 Consumer Perspective Definitions

5.1.1 Perspective 1: Cloud Service Provider (CSP)

Trust Question: “Are the physical hosts and TEE platforms in my fleet running known-good firmware and hardware configurations?”

The CSP operates at the outermost boundary. Their attestation interest is in the integrity of the physical compute hardware and the TEE platform itself. The CSP does not (and should not) have visibility into the workload running inside the TEE – that is the entire point of confidential computing.

Table 8: CSP Attestation Perspective

Aspect	Detail
Evidence Visible	L1 Platform: TPM/SNP hardware attestation, firmware versions, Secure Boot status, TCB versioning
Evidence NOT Visible	L2 and L3: Container images, application behavior, kernel integrity (opaque to CSP)
Actionable Outcomes	Fleet firmware compliance, host-level remediation, SLA reporting on TEE availability
Attestation Delivery	Internal fleet management APIs; not published to external parties
Typical Provider Trust Stance	Provider-Inclusive (the CSP is the provider whose trust anchor types are being authored). See Section 5.3.

5.1.2 Perspective 2: Infrastructure Consumer (IaaS Tenant)

Trust Question: “Is the TEE I provisioned running the virtual machine image I specified, on genuine hardware, with the firmware versions I require?”

The IaaS consumer provisions confidential VMs and needs assurance that the TEE is genuine and that the VM image loaded matches their specification. They have visibility into L1 (through the TEE attestation report) and into the static image state of the VM they deployed, but not necessarily into application-layer containers managed by higher-level consumers.

Table 9: Iaas Consumer Attestation Perspective

Aspect	Detail
Evidence Visible	L1 Platform: Full TEE attestation report, transport binding. L2 Image (partial): VM image hash, hardware-bound loaded binary hashes
Evidence NOT Visible	L2 (container-level) and L3 details managed by PaaS / SaaS operators above them
Actionable Outcomes	VM provisioning verification, KMS access gating, compliance evidence for their cloud deployment
Attestation Delivery	TEE attestation report via Attested Transport Binding; L2 image evidence via Appraiser publisher stream
Typical Provider Trust Stance	Provider-Inclusive (default for tenants operating within their CSP's trust boundary). Provider-Limited or Provider-Independent for regulated workloads or sovereign deployments. See Section 5.3.

5.1.3 Perspective 3: Application Developer / Operator (PaaS / SaaS Provider)

Trust Question: “Are my application containers, AI models, and dependencies running unmodified, and is the runtime behaving as expected?”

The application operator deploys containerized workloads (and potentially AI models) inside the TEE. They have the deepest visibility into L2 and L3 evidence: container image digests, SBOM verification, IMA measurements, eBPF behavioral monitoring, and kernel integrity status. This perspective drives most of the continuous attestation value.

Table 10: Application Operator Attestation Perspective

Aspect	Detail
Evidence Visible	L1 Platform (consumed for baseline trust). L2 Image: Container digests, Sigstore signatures, IMA boot measurements, SBOMs, model weight integrity. L3 Runtime: eBPF behavioral events, kernel integrity results, IMA runtime measurements
Evidence NOT Visible	CSP fleet internals; other tenants’ workloads
Actionable Outcomes	KMS access gating, workload traffic blocking (enforcement gate), SIEM alerting, compliance evidence, AI model provenance verification
Attestation Delivery	Full publisher stream via the deployment’s chosen transport mechanism (see Section 7.1); Attested Transport Binding for direct client verification

Aspect	Detail
Typical Provider Trust Stance	Provider-Limited (default), with Provider-Inclusive or Provider-Independent variants depending on the customer commitments and downstream regulatory obligations the application operator carries. See Section 5.3.

5.1.4 Perspective 4: End User (Client / Data Subject)

Trust Question: “Can I verify that the service I’m about to send my sensitive data to is running inside a genuine, attested TEE – and that it remains healthy throughout my session?”

The end user is the outermost consumer in the trust chain. They do not operate the infrastructure and typically cannot inspect internal attestation layers directly. Their primary mechanism is Attested Transport Binding (Section 8), which allows them to verify the TEE’s hardware identity and bind it to the TLS session before transmitting sensitive data (e.g., AI prompts, personal health information, financial data).

Beyond initial verification, the framework defines a continuous status visibility mechanism (TB-7, Section 8) that enables clients to query the current per-layer attestation status during a session. This bridges the gap between the one-time transport binding check (“Is this a real TEE?”) and the ongoing attestation appraisal (“Is this TEE still healthy?”).

Table 11: End User Attestation Perspective

Aspect	Detail
Evidence Visible	Attestation token via Attested Transport Binding (L1 hardware binding to TLS certificate). Optionally: L2 summary (workload identity), L3 summary (current trust status) via continuous status API
Evidence NOT Visible	Internal evidence bundles, raw eBPF events, kernel memory details
Actionable Outcomes	Proceed / abort connection decision, certificate pinning, Zero Trust session establishment, mid-session trust re-evaluation
Attestation Delivery	Attested Transport Binding (post-handshake or intra-handshake per TB-6 disclosure); continuous status query via attested endpoint (TB-7). Implementations supporting the Full attestation profile (Section 11.1) must provide a continuous status mechanism for End User consumers.
Typical Provider Trust Stance	Provider-Independent (default; the end user has not chosen the provider). Provider-Limited where the SaaS operator publishes its provider relationship and the end user accepts that scope. See Section 5.3.

5.2 Perspective Visibility Matrix

The following matrix summarizes which evidence elements are visible to each consumer perspective:

Table 12: Perspective Visibility Matrix

Evidence Element	CSP	IaaS Tenant	App Operator	End User
Hardware Attestation Report	Full	Full	Full (via Measurement Gatherer)	Via Transport Binding
Firmware / TCB Version	Full	Full	Full	Summary
GPU Attestation Report	-	-	Full	Via Transport Binding (if composite)
VM Image Hash	-	Full	Full	-
Container Image Digests	-	-	Full	-
SBOM / Sigstore Signatures	-	-	Full	-
IMA Boot Measurements	-	Partial	Full	-
IMA Runtime Measurements	-	-	Full	-
eBPF Behavioral Events	-	-	Full	-
Kernel Integrity Monitor (KIM)	-	-	Full	-
Overall Trust Status	-	Signed verdict	Signed verdict	Via Transport Binding / Status API

The consumer perspectives define evidence visibility and actionable scope, not independent verdict computation. There is one overall trust verdict per Target Machine, derived from the decision matrix in Section 4.4. All perspectives observe the same underlying evidence at their respective visibility levels. If two perspectives appear to reach different conclusions, this reflects a visibility boundary where one perspective sees evidence the other cannot, rather than a conflict in the underlying verdict. The verdict is singular and authoritative; it is the interpretation and visibility of supporting evidence that varies across perspectives.

5.3 Provider Trust Stance

Each consumer perspective in Section 5.1 takes a stance on which trust anchor types (Section 4.1.3) it accepts as evidence. The framework names three Provider Trust Stances so that implementations can declare their stance and so that the same evidence stream can produce appropriate verdicts at different consumer tiers. The Provider Trust Stance is a property of the consumer's policy, not of the evidence itself, and is independent of the Attested Transport Binding requirements in Section 8.

Stance 1: Provider-Inclusive. The consumer accepts Type A, Type B, and Type C trust anchors. This is the typical stance for the Cloud Service Provider perspective itself, for Infrastructure Consumers operating within their CSP's trust boundary, and for Application Operators whose threat model treats the CSP as cooperating infrastructure. Provider-Inclusive is the stance that enables the broadest reuse of provider-operated attestation services and is the entry point for cloud adoption of the framework.

Stance 2: Provider-Limited. The consumer is selective about which operating services are acceptable in the verification path. A common variant accepts Type A and Type B but excludes Type C. Another variant accepts Type A and a specific Type C for one component while excluding others. The stance is implementation-declared and policy-driven and is appropriate where regulatory, contractual, or sovereign requirements distinguish between classes of operating service.

Stance 3: Provider-Independent. The consumer accepts only Type A trust anchors. This is the typical stance for End Users who have not chosen the CSP themselves, for highly regulated workloads where the CSP cannot be part of the TCB, and for cross-jurisdictional deployments where the CSP is in a different trust domain than the data subject.

The stance does not change what evidence is collected; it changes which evidence is considered acceptable when composing the trust verdict. Implementations should be capable of producing evidence streams that serve multiple stances simultaneously, allowing the same Target Machine to satisfy a Provider-Inclusive Infrastructure Consumer and a Provider-Independent End User from a single attestation cycle.

The Provider Trust Stance composes orthogonally with the Attestation Depth Profile (Section 11.1). A relying party declares both: a Depth Profile that determines which evidence layers are active, and a Provider Trust Stance that determines which trust anchor compositions are acceptable. The two dimensions can be combined into a single declaration such as "Full Depth, Provider-Inclusive" or "Workload Depth, Provider-Independent."

6. AI Workload Attestation Controls

AI and machine learning workloads present unique attestation challenges that do not arise in traditional application deployments. This section defines how AI-specific integrity concerns map into the framework's existing three-layer evidence model. These controls are not a separate "AI layer" – they are extensions and specializations within L2 and L3 that address the distinct characteristics of AI systems.

Design Principle: AI attestation controls are integrated into the existing layer structure, not added as a separate layer. This preserves the framework's architectural simplicity while ensuring AI-specific threats are addressed at the appropriate verification boundary.

The AI-specific threat landscape for machine learning systems is catalogued by MITRE ATLAS (Adversarial Threat Landscape for AI Systems), which provides a knowledge base of adversary tactics and techniques against AI. The controls defined in this section address a subset of ATLAS-relevant threats, specifically those that are detectable and mitigable through the framework's attestation evidence model. The full ATLAS matrix should be consulted for comprehensive AI threat coverage, recognizing that many ATLAS techniques (e.g., adversarial prompt injection, training data poisoning at the data pipeline level) operate at the application or data layer and fall outside the scope of infrastructure attestation.

6.1 AI-Specific Challenges in Confidential Computing

AI workloads differ from traditional applications in several ways relevant to attestation:

- **Large Artifact Size:** Model weight files can exceed 100 Gigabyte (GB). Full IMA-style hashing at boot would impose unacceptable latency. The framework distinguishes between executable code (hashed via IMA) and data artifacts such as model weights (verified via authenticated encryption).

- **Data Provenance:** Retrieval-Augmented Generation (RAG) pipelines ingest external data at runtime. Verifying the integrity and provenance of these inputs – vector databases, document corpora, embedding indices – is a security-relevant concern that falls outside traditional binary attestation.

- **Model Drift and Behavioral Change:** Even without code modification, an AI model's outputs can shift due to poisoned training data, manipulated fine-tuning, or adversarial prompt injection. Detecting such changes requires monitoring "security-relevant behavioral changes" beyond traditional syscall-level eBPF monitoring.

- **Prompt Guardrail Integrity:** AI deployments often include safety guardrails (content filters, system prompt enforcement). The integrity of these guardrail configurations is a security-relevant property that should be attestable.

- **Interpreted and Dynamic Code:** AI inference pipelines are frequently implemented in interpreted languages (Python, Node.js) where code may be dynamically loaded, Just-In-Time (JIT)-compiled, or fetched at runtime. IMA-based measurements are most effective for compiled executables and statically-deployed files, dynamically loaded scripts and runtime-generated code may not be captured. The AI controls in this section should be supplemented with L3.2 behavioral monitoring for interpreted workloads.

6.2 AI Controls Mapped to Evidence Layers

Table 13: AI Controls Mapped to Evidence Layers

AI Control	Framework Layer	Evidence Element	Verification Method
AI-L2.1: Model Weight Integrity	L2 (Image)	Digest of encrypted model weight files at rest, verified during decryption	Authenticated Encryption (AEAD) integrity check during KMS-gated decryption. Model weights treated as data, not executable code.
AI-L2.2: Model Provenance and Signing	L2 (Image)	Cryptographic signature over model artifact manifest (weights, tokenizer, config)	Sigstore/Cosign signature verification against the model publisher's signing key. Included in SBOM.
AI-L2.3: RAG Data Source Registration	L2 (Image)	Signed manifest of authorized RAG data sources (vector DB endpoints, document corpus hashes)	Manifest included in Golden Baseline. Connections to unregistered data sources flagged as L3 violations. The RAG data source manifest follows the composite baseline model (Section 4.2.2): platform-defined data source restrictions cannot be overridden by tenant-supplied manifests, but tenants may authorize additional sources within those bounds.
AI-L2.4: Guardrail Configuration Integrity	L2 (Image)	Hash of prompt guardrail configuration files (system prompts, filter rules, safety policies)	IMA measurement of guardrail config files at load time. Hash included in L2 baseline. Guardrail logic implemented in application source code (rather than external configuration files) is covered by the container image digest (L2) rather than individual IMA measurement. Implementations SHOULD externalize guardrail definitions into dedicated configuration files to enable granular IMA measurement.

AI Control	Framework Layer	Evidence Element	Verification Method
AI-L3.1: Inference Pipeline Behavioral Profile	L3.2 (Application)	eBPF behavioral profile for inference processes: expected syscalls, network connections. *NOTE: GPU-specific behavioral evidence (e.g., GPU memory access patterns) depends on GPU vendor telemetry capabilities, which are evolving and not yet standardized.*	Behavioral monitoring via eBPF sensors. Deviations trigger ALERT or FAIL per policy.
AI-L3.2: Data Provenance Runtime Validation	L3.2 (Application)	Runtime verification that RAG queries are directed only to registered data sources	eBPF network hooks validate egress connections against the AI-L2.3 manifest. Unauthorized connections blocked / alerted.
AI-L3.3: Model Drift Detection Hooks	L3.2 (Application)	Integration point for external model monitoring services that detect statistical drift in model outputs	The framework identifies model drift detection as an integration point for specialized external monitoring tools. The evidence schema for drift metrics will be defined in a future revision. Implementations may report externally generated drift signals as L3.2 evidence.

6.2.1 Model Weight Integrity: The AEAD Approach

Model weights are treated as data artifacts rather than executable code. Integrity is verified through authenticated encryption during decryption, rather than IMA-style hashing at boot. This approach provides equivalent integrity assurance while avoiding the boot-time latency penalty of hashing large model files. The resulting digest should be recorded as attestation evidence.

7. Publisher Protocol and Anti-Replay Mechanisms

The Publisher role is the interface between the in-boundary attestation system and the external Enterprise Validator. It streams signed attestation results while providing strong guarantees against replay, withholding, and modification attacks.

7.1 Protocol Specification

The Publisher role requires a streaming or event-driven transport capable of delivering signed attestation results from the in-boundary Appraiser to one or more Enterprise Validators with predictable latency, ordering, and integrity properties. The framework does not mandate a single publication transport protocol. YANG Push (RFC 8641) over RESTful Network Configuration Protocol (RESTCONF) with Server-Sent Events remains a well-suited mechanism – it provides standards-based subscription management (periodic and on-change), structured YANG-modeled payloads, and alignment with IETF network management conventions – but it is not the only viable choice. Deployments should select a publication transport that matches their existing infrastructure, latency requirements, and operational expertise.

The following mechanisms are architecturally compatible with the Publisher role defined in this framework:

- **YANG Push (RFC 8641) over RESTCONF/SSE:** Standards-track subscription protocol with IETF pedigree, structured data modeling via YANG, and native support for periodic and on-change notification modes. Well-suited to environments already operating Network Configuration Protocol (NETCONF)/RESTCONF-based management planes. Downside: limited adoption outside traditional network management tooling; SSE is unidirectional (server-to-client only), which constrains interactive query patterns.

- **gRPC Streaming**: Bidirectional streaming over Hypertext Transfer Protocol (HTTP)/2 with strong payload typing via Protocol Buffers. Native to cloud-native and Kubernetes-based orchestration environments, with mature load balancing, health checking, and deadline propagation. Downside: requires HTTP/2 end-to-end, which complicates traversal of certain proxies and legacy middleboxes; less established in traditional enterprise network management tooling.

- **Message Queuing Telemetry Transport (MQTT) v5:** Lightweight publish-subscribe protocol with configurable Quality of Service levels (QoS 0: at-most-once, QoS 1: at-least-once, QoS 2: exactly-once). Widely deployed in IoT, ICS/OT, and edge environments where bandwidth is constrained and connectivity is intermittent. Downside: requires a message broker as infrastructure, which introduces a dependency outside the TEE trust boundary; less common in cloud-native enterprise stacks.

- **CloudEvents over HTTP/SSE or WebSocket:** Cloud Native Computing Foundation (CNCF)-standard envelope format for describing event data in a transport-agnostic, JavaScript Object Notation (JSON)-based schema. Firewall-friendly over standard HTTP ports, with broad tooling support across serverless and event-driven architectures. Downside: CloudEvents defines the event envelope, not delivery guarantees – ordering, exactly-once semantics, and durability must be layered by the implementation or an underlying transport.

- **Apache Kafka / Event Streaming Platforms:** Durable, ordered, replayable event logs with enterprise-grade partitioning, configurable retention, and consumer group semantics. Well-suited to deployments requiring long-term evidence retention, audit replay, or fan-out to multiple downstream consumers. Downside: significant infrastructure dependency (broker cluster, ZooKeeper / KRaft); higher latency overhead for small, single-publisher payloads; operationally heavy for deployments with a small number of Target Machines.

Regardless of the transport mechanism chosen, the anti-replay, freshness, event signing, and dead man's switch requirements defined in Section 7.2 remain normative. Every published event must carry a monotonic epoch identifier (ID), a cryptographic signature bound to the Publisher's attestation-derived signing key, and a timestamp sufficient for staleness evaluation by the Enterprise Validator. Implementations must disclose the chosen publication transport in the attestation metadata so that Enterprise Validators can configure their ingestion pipelines accordingly.

7.2 Anti-Replay and Freshness

Table 14: Anti-Replay and Freshness Mechanisms

Mechanism	Description	Threat Addressed
Monotonic Epoch IDs	Unit64 counter per Target Machine, incremented per event. Validators reject any event with an epoch_id $\leqslant$ the last seen value	Replay attacks, event reordering
Event Signatures	Event Signatures Digital signature over (target_id + status + epoch_id + timestamp) using the Publisher's attestation-bound signing key and a NIST-standardized post-quantum signature algorithm (e.g., ML-DSA per Federal Information Processing Standards (FIPS) 204).	Event modification, fake event injection
Query Nonce Binding	REST status queries accept an optional nonce from the Validator. The nonce is echoed and signed in the response.	Cached response replay
Dead Man’s Switch	If a Target Machine stops reporting evidence within the dead man’s window (see Section 7.4), the Publisher automatically publishes a FAIL status and triggers trust revocation.	Publisher silence, network isolation attacks

The Publisher's signing key used for event signatures must be generated inside a TEE or equivalent protected environment using a NIST-standardized post-quantum signature algorithm (e.g., ML-DSA per FIPS 204). "Attestation-bound" means the key's public key hash is embedded in verifiable attestation evidence (analogous to the TB-1 key commitment for transport binding), enabling Enterprise Validators to confirm the signing key is held in a protected environment before accepting signed events. This key may be the same key used for TLS termination at the Publisher's endpoint, provided it satisfies both the transport binding and event signing requirements. The signing key should not rotate mid-session; rotation occurs naturally on Publisher restart, at which point a new attestation binding must be established and verified by Validators before event publication resumes.

7.2.1 Reboot and Session Continuity

A reboot, crash recovery, or TEE re-initialization constitutes a new attestation session and resets the monotonic epoch counter. The Validator must not inherit trust from the previous session. All trust state associated with the old session must be invalidated, and the Validator must require a complete re-attestation cycle before accepting the new session as trusted.

7.3 The Lying Publisher Problem

A compromised Publisher could attempt to suppress FAIL notifications, replay stale PASS states, or modify evidence. The framework addresses this through three complementary mechanisms:

- **Integrity:** The Publisher forwards TEE-signed artifacts. It cannot forge attestation evidence because it does not possess the TEE’s signing keys.

- **Anti-Silence:** The Enterprise Validator enforces a heartbeat policy. If the Publisher fails to send a signed keep-alive or update within window T, the Validator defaults to “Untrusted.”

- **Public Auditability:** Appraisal results are logged to a public, append-only transparency log (e.g., Sigstore Rekor). Any discrepancy between the Publisher’s stream and the transparency log proves the Publisher lied – publicly and irrefutably.

7.4 Attestation Interval Policy

The attestation reporting interval is a policy parameter that determines how frequently the Measurement Gatherer collects evidence and the Appraiser evaluates it. The interval directly impacts the granularity of continuous attestation and the maximum staleness window for compliance assertions.

Implementations should support on-demand attestation in addition to periodic attestation, whereby an Enterprise Validator or another authorized relying party triggers an immediate, out-of-cycle evidence collection and appraisal. Representative use cases include reaction to threat intelligence alerts, fulfilment of compliance audit requests, support for incident response actions, and security posture queries issued by orchestration or fleet management systems. On-demand attestation produces the same evidence bundle and follows the same appraisal logic as periodic attestation; it bypasses the configured reporting interval timer without changing evidence composition or verdict semantics.

The framework defines the following requirements and recommendations for the attestation interval:

- **Policy-Driven:** The reporting interval must be configurable, not hardcoded.
- **Metadata Disclosure:** The current reporting interval must be included in the attestation metadata published by the Publisher, enabling Enterprise Validators to assess evidence freshness and calibrate their own staleness thresholds.
- **Dead Man’s Switch Calibration:** The dead man's switch window must be calibrated to the reporting interval. The recommended window is 2× the configured reporting interval.
- **Compliance Assertion Time to Live (TTL):** Compliance assertion TTLs should be calibrated to the reporting interval to ensure assertions do not outlive their supporting evidence.
- **On-Demand:** Implementations should support on-demand attestation so that an Enterprise Validator or authorized relying party may request an immediate attestation cycle, the on-demand path shall produce the same evidence bundle and shall apply the same appraisal logic as periodic attestation, differing only in that the cycle is not gated on the interval timer.
- **Compliance Assertion Revocation:** A FAIL verdict from any attestation layer immediately invalidates all outstanding compliance assertions for that Target Machine, regardless of their remaining TTL. Enterprise Validators must treat a FAIL event as superseding any previously issued assertion. This ensures that a valid compliance assertion and a FAIL verdict cannot coexist.
- **ICS/OT Timing Constraints:** Industrial control system environments may impose deterministic timing requirements on all software executing within the control loop. In these deployments, attestation evidence collection and appraisal cycles must not interfere with control loop latency - for example, eBPF evidence gathering should be scheduled outside the real-time task window or throttled to avoid pre-empting time-critical process control threads. The attestation interval should be calibrated to the control cycle period: intervals shorter than the control

cycle provide no additional security value and risk disrupting deterministic execution, while intervals significantly longer than the control cycle increase the staleness window for safety-critical workloads. Deployments should document their control loop timing constraints and the attestation interval rationale in the attestation metadata.

8. Attested Transport Binding

8.1 Problem Statement

Standard TLS authenticates a server's identity through certificate authorities but provides no assurance about the execution environment terminating the connection. A valid TLS certificate could be served by a compromised host, a relay, or a machine that is not running inside a TEE at all.

Attested Transport Binding is the cryptographic property that links a TEE's hardware attestation evidence to a specific transport session, ensuring that the attested TEE and the TLS endpoint are the same entity. Without this binding, attestation evidence and encrypted transport are independent – an adversary can present valid attestation from a genuine TEE while intercepting the actual traffic at a different endpoint.

This threat – the relay attack – is the primary motivation for requiring transport binding in any deployment where a remote party (the End User perspective, Section 5) makes trust decisions based on attestation evidence.

8.2 Requirements

Conforming implementations must satisfy the following transport binding requirements. These requirements are platform-neutral and apply regardless of the underlying TEE technology (AMD SEV-SNP, Intel TDX, Arm CCA, NVIDIA CC), the cloud provider (Azure, Amazon Web Services (AWS), Google Cloud Platform (GCP), on-premise), or the attestation token format (platform-specific JWTs, raw hardware reports, NRAS EAT tokens).

- **TB-1. Key Commitment:** The attestation evidence must contain a cryptographic commitment to the TLS session's public key. This commitment binds the attestation to a specific key pair. The mechanism by which the commitment is embedded depends on the platform's attestation format (e.g., a nonce field, a user-data field in a hardware report, or a claim in a signed token). The specific field and encoding are implementation-defined, but the binding must be verifiable by the client.

Note on key commitment scope: TB-1 binds attestation evidence to the TLS public key, not directly to the derived session key. The session key is computed during the TLS handshake from parameters that are authenticated by the committed public key. This derivation chain ensures that only the party holding the attested TLS private key can complete the handshake and derive session keys for that connection. The binding is therefore transitively sound: committing to the public key in the attestation evidence attests the entity that can derive session keys from it. Existing Remotely Attested TLS (RA-TLS) implementations, including those analysed by the Confidential Computer Consortium (CCC) Attestation (Signature / Special Interest Group) SIG, embed the TLS

public key hash in the attestation report's user-data or nonce field, which the client verifies against the certificate presented during the handshake.

- **TB-2. Freshness:** The attestation evidence must include a freshness guarantee to prevent replay of stale attestation. Acceptable freshness mechanisms include a client-supplied nonce echoed in the signed evidence, a verifiable short-lived timestamp from a trusted time source, or a hardware-attested monotonic counter. The choice of freshness mechanism is implementation-defined. Implementations may provide tiered freshness guarantees, where higher tiers provide stronger per-session binding (e.g., incorporating TLS Exporter values) at the cost of additional latency. Such tiering allows clients to select the appropriate freshness level for their risk profile.

- **TB-3. Client Verification:** The client must verify that the cryptographic commitment in the attestation evidence matches the TLS certificate or public key presented during the session. The client must reject the session if the binding does not match, if the attestation evidence fails verification, or if the freshness guarantee is not satisfied.

- **TB-4. Certificate Pinning:** After successful verification, the client should pin the TLS certificate for the duration of the session to prevent mid-session certificate substitution.

- **TB-5. Key Protection Disclosure:** Implementations must disclose how the TLS private key is protected within the TEE. The security of transport binding depends directly on the key's protection: a key held in hardware-isolated storage inaccessible to the guest kernel provides stronger guarantees than a key held in guest memory. Implementations must document their key storage model and the residual risks associated with it, enabling relying parties to make informed trust decisions.

- **TB-6. Binding Mechanism Disclosure:** Implementations must disclose whether attestation binding occurs post-handshake (attestation delivered after TLS connection establishment) or intra-handshake (attestation embedded within TLS handshake messages). Formal analysis has identified relay attack vectors in certain intra-handshake approaches. Implementations using intra-handshake binding must document the specific relay mitigations employed and the deployment constraints under which those mitigations are effective.

- **TB-7. Continuous Status Visibility (Informative):** After initial transport binding verification, the attested endpoint should provide a mechanism for clients to query the current attestation status. The mechanism, trust model, and authentication requirements for this endpoint are implementation-defined.

NOTE: For post-quantum transport, transport sessions should employ quantum-resistant key exchange mechanisms using a NIST-standardized post-quantum key

encapsulation mechanism (e.g., Machine Learning-Key Encapsulation Mechanism (ML-KEM) per FIPS 203, or a hybrid such as SecP256r1MLKEM768 during transition). This mitigates harvest-now-decrypt-later attacks on attestation-bound sessions. Organizations subject to Commercial National Security Algorithm Suite (CNSA) 2.0 requirements should evaluate quantum-resistant TLS deployment timelines, noting that CNSA 2.0 mandates quantum-resistant key establishment as early as 2030 for non-national-security systems.

Informative Note on TLS Session Resumption: TLS 1.3 supports session resumption via pre-shared keys (PSK) and 0-Round Trip Time (0-RTT) early data (RFC 8446 §2.3, §8). These mechanisms interact with transport binding as follows:

- *0-RTT early data is replayable by design and provides no forward secrecy. An attacker who captures a ClientHello containing early data can re-send it to the same or – in load-balanced deployments where session ticket encryption keys are shared across a pool of Target Machines – a different endpoint. This directly conflicts with TB-2 freshness and, in pooled deployments behind L4 load balancers (Topology Class 1), enables cross-TEE replay where attested data intended for one Target Machine is accepted by another. Implementations must disable 0-RTT early data on attestation-bound endpoints or must reject early data and require a full handshake when attestation evidence is requested.*

- *PSK resumption with key exchange (`psk_dhe_ke` mode) establishes fresh session keys with forward secrecy but does not re-establish TB-2 attestation freshness – the client operates on attestation evidence from the original handshake, which may be arbitrarily stale if the attestation interval has elapsed. Implementations should adopt one or more of the following mitigations: disable session tickets on attestation-bound endpoints; enforce a maximum session ticket lifetime no greater than the configured attestation reporting interval (Section 7.4); or require clients to re-verify attestation freshness via the TB-7 status query mechanism before sending application data on a resumed session. The chosen policy must be disclosed in the attestation metadata.*

- *For Topology Class 2 deployments that implement application-layer attestation binding, TLS session resumption at the intermediary does not affect the application-layer attestation channel, which maintains its own key state independent of the TLS session. However, TLS resumption between the intermediary and the Target Machine is subject to the same constraints defined above and must be configured accordingly on that path segment.*

8.3 Informative Guidance on Key Protection

The strength of transport binding depends on how the TLS private key is protected within the TEE. Implementations should document their key storage model, including whether the key is accessible to the guest kernel and the isolation

mechanisms employed. Stronger key isolation reduces the residual risk from transient kernel compromises but may introduce performance trade-offs. The specific mechanisms for key isolation vary by TEE platform and are evolving.

8.4 Cross-Platform Applicability

Transport binding as defined by requirements TB-1 through TB-6 is achievable across the full range of current confidential computing platforms. The attestation token format, the field used for key commitment, and the certificate chain verification procedure differ by platform, but the abstract requirements are uniformly applicable. Implementations targeting multiple platforms will need platform-specific attestation and verification logic, but the transport binding properties and client-side verification requirements remain constant.

Table 15: Cross-Platform Applicability of Transport Binding

Platform Dimension	Varies by Implementation	Constant Across Implementations
TEE Technology	AMD SEV-SNP, Intel TDX, Arm CCA, NVIDIA CC	TB-1 through TB-6 requirements
Cloud Provider	GCP, Azure, AWS, on-premise	Key commitment and freshness properties
Attestation Token Format	Platform JWTs, raw hardware reports, NRAS EAT tokens	Client verification obligation
Key Commitment Field	eat_nonce, REPORT_DATA, platform-specific claims	Binding must be verifiable by client
Key Storage Model	Hardware-isolated, in-memory	Disclosure obligation (TB-5)
Certificate Chain	Platform-specific (VCEK, Intel QE, Arm CCA, NRAS)	Chain must root to hardware trust anchor
Trust Anchor Composition	Type A (silicon-rooted), Type B (provider-mediated), Type C (vendor-service-mediated), or mixed across evidence elements	Per-element disclosure obligation in attestation metadata; see Section 4.1.3

8.5 Deployment Constraints: Transport Intermediaries

Attested Transport Binding as defined by TB-1 through TB-6 requires an end-to-end TLS session between the client and the TEE. Transport intermediaries that terminate and re-establish TLS – including L7 load balancers, Web Application Firewalls (WAFs), API gateways, CDN edge nodes, and service mesh sidecars with TLS origination – break the key commitment binding because the client's TLS session terminates at the intermediary, not at the TEE endpoint.

This is a fundamental architectural constraint, not an implementation limitation. The security property that TB-1 provides (the attestation evidence contains a commitment to the TLS public key the client connected to) cannot hold when the

TLS public key the client sees belongs to the intermediary rather than the TEE. Because these intermediaries are standard components of production architectures, this constraint affects the majority of real-world deployments. Implementers must understand the topology implications before evaluating compliance with TB-1 through TB-6.

The framework identifies the following deployment patterns and their transport binding implications:

- **Direct Connection (No Intermediary):** Fully compatible with all transport binding requirements.

- **L4 (TCP) Pass-Through Load Balancers:** TCP-level load balancers that forward connections without TLS termination preserve the end-to-end TLS session and are fully compatible with TB-1 through TB-6. This is the recommended load balancing approach for deployments requiring transport binding. L4 load balancers sacrifice application-layer routing capabilities (Uniform Resource Locator (URL)-based routing, header inspection, request-level WAF policies) in exchange for preserving the end-to-end cryptographic channel. For workloads where the load balancer's role is distributing connections across a pool of Target Machines common in AI inference deployments, this trade-off is acceptable, and the resulting transport binding is the strongest achievable. Deployments that do not require L7 routing features should evaluate L4 pass-through as the simplest path to full transport binding compliance.

- **L7 Load Balancers with TLS Termination:** These intermediaries break transport binding. Deployments using L7 load balancers must disclose this constraint, including which transport binding requirements are satisfied end-to-end (client to Target Machine) and which are satisfied only on the intermediary-to-Target-Machine segment. Deployments should employ one or more of the following mitigations:

    - *(a) Application-layer attestation binding:* The client and the Target Machine negotiate an inner cryptographic channel (e.g., application-layer key agreement using Elliptic Curve Diffie-Hellman (ECDH) with attestation token binding) that is independent of the TLS session. The attestation evidence contains a commitment to the application-layer key rather than the TLS key, and the client verifies this commitment after the inner channel is established. The outer TLS session (terminated at the intermediary) provides transport encryption as defense-in-depth; the inner channel provides end-to-end attestation binding. This pattern survives TLS re-establishment at the intermediary and preserves full attestation properties for the client. When this mitigation is employed, TB-1 key commitment may bind to the application-layer key rather than the TLS key, provided

the binding is documented and verifiable. Working implementations exist in production deployments but are not yet standardized.

- *(b) Intermediary attestation chain:* Where the L7 intermediary is operated within the deployment's trust boundary (e.g., an organization's own load balancer, not a third-party CDN), the deployment may establish a chain of custody by attesting the intermediary itself and binding its identity to the intermediary-to-Target-Machine TLS session. This provides a two-hop binding: client-to-intermediary (standard TLS) and intermediary-to-Target-Machine (attested TLS). The residual trust assumption is that the intermediary is operating correctly, which is a weaker property than end-to-end binding but is auditable and verifiable.

- *(c) L4 migration:* Deployments that currently use L7 load balancers should evaluate whether migration to L4 (TCP pass-through) load balancing is operationally feasible. Many AI inference and confidential computing workloads do not require L7 routing features, and the migration eliminates the transport binding constraint entirely.

- **Hybrid Topologies:** Some deployments serve multiple consumer perspectives (Section 5) through different network paths. For example, internal service-to-service communication may use direct connections or L4 load balancing (achieving full transport binding for the Application Provider perspective), while end-user traffic arrives through a CDN or L7 load balancer (where transport binding holds only to the intermediary). Hybrid deployments must document each network path, the consumer perspective it serves, and the transport binding properties achieved on that path. Enterprise Validators should evaluate transport binding on a per-path basis rather than requiring a single uniform binding model across all access patterns.

Implementations must disclose which topology applies to their deployment, which mitigations are employed, and what residual risks remain. Enterprise Validators should account for the transport binding model when evaluating the attestation strength of a deployment.

NOTE: The CCC Attestation SIG formally proved that certain intra-handshake attestation mechanisms are vulnerable to relay attacks. The L7 load balancer constraint is a distinct and complementary limitation: post-handshake attestation is secure against relay attacks (as proven) but requires end-to-end TLS connectivity, while intra-handshake approaches face relay vulnerabilities regardless of infrastructure topology. Both constraints should be considered when selecting an attestation binding approach for a given deployment.

9. Threat Model (EMB3D + Cloud Overlay)

This section is informative. It motivates the normative requirements defined elsewhere in the framework but does not itself impose additional requirements on implementations.

The threat model draws upon MITRE EMB3D for hardware and firmware threats relevant to Layer 1 and applies a Cloud Overlay to address threats specific to cloud confidential computing environments. The Cloud Overlay excludes physical tampering (CSP responsibility) and adds cloud orchestration and publisher protocol threats. The framework leveraged EMB3D and established cloud threat models to identify the threats most pertinent to confidential computing; the specific enumeration of adversary capabilities and mitigation mechanisms in the tables below is what matters for implementers.

9.1 Adversary Model

Table 16: Adversary Model

Adversary	Description	Assumed Capabilities
System Operator (The Host)	Malicious insider at the CSP, or attacker with privileged infrastructure access	Full control over hypervisor, host OS, network fabric, disk storage. Can stop/start VMs, inspect unencrypted memory outside TEE, manipulate I/O rings.
Shared Platform Tenant (The Neighbor)	Malicious actor on the same physical host	Side-channel attacks (Spectre/Meltdown), cross-container exploits via shared kernel structures
Malicious Tenant Insider (The Insider)	User with valid deployment credentials who subverts governance	Deploying compromised workloads, injecting malware into OS images, modifying kernel modules
External Actor	Traditional network-based attacker	Exploitation via public interfaces, supply chain injection, MITM attacks

9.2 Threat Matrix

Table 17: Threat Matrix

Layer / Domain	Threat	Mitigation
L1: Platform	Firmware Compromise	Hardware Root of Trust (TPM/SNP) verifies boot chain against Golden Policy
L1: Platform	Side-Channels (Spectre/Meltdown)	TCB version check enforces minimum microcode patch levels. Residual risk: zero-day side-channels.
L1: Platform	Firmware Rollback	min_tcb_version enforcement rejects old firmware/kernel versions

Layer / Domain	Threat	Mitigation
L1-GPU: Platform	GPU Firmware Compromise	GPU vendor attestation report verifies GPU firmware and driver versions against baseline. Shared nonce binding prevents cross-device replay.
L2: Image	Supply Chain Injection	IMA + Sigstore signature verification against trusted CI/CD root key
L2: Image	Malicious Insider	Composite Baseline: Vendor reference overrides tenant allowlist to prevent insider from authorizing compromised kernel
L2: Image	TOCTOU Swap	Attestation token binds specific image digest to the released session via transport binding key commitment (TB-1)
L3: Runtime	Kernel Modification (Rootkit)	Kernel Integrity Monitor validates memory structures against golden image (L3.1)
L3: Runtime	Application RCE	eBPF behavioral containment detects post-exploit behavior (L3.2). Does not prevent initial bug.
L3: Runtime	Network Exfiltration	eBPF egress filtering blocks unauthorized connections. Cannot prevent exfiltration via allowed side-channels.
Orchestration	Hypervisor Tampering	TLS encryption + eBPF packet validation. DoS by host is out of scope.
Orchestration	Malicious Sidecar Injection	L2 Composite Baseline: all container digests inside the TEE are measured and compared against the Golden Baseline. An injected sidecar whose digest is not in the allowlist triggers an immediate L2 FAIL.
Orchestration	Workload Rescheduling to Unattested Node	L1 Platform Attestation: the rescheduled workload fails L1 verification because the target node does not produce a valid hardware attestation report. State-gated I/O prevents key release.
Orchestration	Network Policy Manipulation	L3.2 eBPF egress filtering operates inside the TEE, independent of external network policy. Network policy changes outside the TEE cannot override in-TEE eBPF rules. Residual risk: traffic manipulation between the TEE boundary and external endpoints is not detectable by L3.2.
Protocol	Lying Publisher	TEE-signed events + heartbeat dead man’s switch + transparency log
Protocol	Replay Attack	Monotonic epoch IDs + challenge-response nonces
Protocol	Relay / MITM	Attested Transport Binding with key commitment (TB-1) and freshness (TB-2). Formal analysis has identified relay vectors in certain intra-handshake approaches (see TB-6).

9.3 Out of Scope and Residual Risks

Table 18: Out-of-Scope Items and Residual Risks

Category	Rationale
Availability / Denial of Service	The hypervisor can terminate the VM, starve CPU resources, or drop network packets. Confidential computing guarantees confidentiality and integrity, not availability.
Residual Side-Channels	Zero-day or unpatched microarchitectural side-channels are an inherent residual risk of multi-tenant shared silicon.
Intrinsic Workload Vulnerabilities	The framework does not make unsafe code memory-safe. It detects and contains consequences of exploitation, not the initial bug.
Data-Only / Logic Attacks	Heap/stack manipulation without control-flow alteration (e.g., changing an isAdmin flag, SQL injection) does not trigger eBPF/kernel signals. These are application-layer failures.
Physical Attacks	Chip decapping, probing, and glitching are managed by CSP physical security controls.

9.4 ATT&CK for ICS Overlay (Informative)

For deployments in OT/ICS environments – such as TEE-protected protocol gateways, historian servers, or supervisory edge controllers – the framework's threat model can be extended with MITRE Adversarial Tactics, Techniques, and Common Knowledge (ATT&CK) for Industrial Control Systems. ATT&CK for ICS catalogues adversary tactics and techniques observed in real-world attacks against industrial control systems and provides a complementary lens to EMB3D: where EMB3D addresses hardware and firmware threats at the device level, ATT&CK for ICS describes adversary behaviours at the operational and network levels that are characteristic of OT environments.

The following table maps selected ATT&CK for ICS techniques to the framework's evidence layers, illustrating how continuous attestation mitigates OT-specific attack patterns.

Table 19: ATT&CK for ICS Overlay

ATT&CK for ICS Technique	Description	Framework Mitigation
T0859 – Valid Accounts	An adversary uses compromised credentials to deploy unauthorized control logic or modified firmware to OT devices through the TEE-protected gateway. Valid credentials alone are insufficient to subvert a continuously attested workload.	L2 Composite Baseline prevents execution of unauthorized code regardless of credential validity. The two-tier baseline model (Section 4.2.2) ensures that platform-level reference values cannot be overridden by a compromised operator account, and any code not present in the allowlist triggers an immediate L2 FAIL.
T0831 – Manipulation of Control	An adversary manipulates control system commands or parameters passing through the TEE to cause	L3.2 behavioral monitoring detects anomalous process behavior and unauthorized network connections in

ATT&CK for ICS Technique	Description	Framework Mitigation
	unsafe physical outcomes (e.g., altering setpoints, disabling safety interlocks).	OT gateway workloads. Deviations from the learned behavioral profile – such as unexpected outbound connections to Programmable Locig Controllers (PLCs) or atypical command patterns – trigger ALERT or FAIL per policy.
T0857 – System Firmware	An adversary modifies firmware on the TEE platform itself or uses a compromised TEE gateway to push tampered firmware to connected OT devices.	L1 Platform attestation enforces `min_tcb_version` and firmware integrity verification via the hardware Root of Trust. Firmware rollback or modification is detected at the next attestation cycle and triggers immediate trust revocation.
T0886 – Remote Services	An adversary exploits remote access protocols (RDP, SSH, VNC, OPC UA) to gain access to TEE-protected OT gateways or to pivot from the gateway into the control network.	Attested Transport Binding (TB-1 through TB-6) prevents relay and man-in-the-middle attacks on the transport layer. L3.2 egress filtering detects unauthorized outbound connections from the TEE, limiting an adversary's ability to use the gateway as a pivot point.

This overlay is informative. The four techniques above were selected because they represent attack patterns where TEE-based attestation provides direct, measurable mitigation. The full ATT&CK for ICS matrix (14 tactics, 80+ techniques) should be consulted for comprehensive OT threat coverage. Deployments in regulated OT environments (e.g., Noth American Electric Reliability Corporation (NERC) Critical Infrastructure Protection (CIP), International Electrotechnical Commission (IEC) 62443) should perform a complete mapping of their applicable ATT&CK for ICS techniques against the framework's evidence layers as part of their risk assessment.

10. Standards Alignment

Table 20: Standards Alignment

Standard	Alignment
IETF RFC 9334 (RATS Architecture)	Full role coverage via Split Verifier model. Evidence format: EAT with eat_nonce. Attestation Results: Attestation Results for Secure Interactions (AR4SI) alignment planned. Reference Values: CoRIM migration planned.
IETF RFC 8641 (YANG Push)	Publisher protocol specification for continuous notification streaming.
MITRE EMB3D	Layer 1 threat taxonomy informing hardware and firmware threat identification, with Cloud Overlay for hypervisor and orchestration threats.
MITRE Embedded Systems Threat Matrix (ESTM)	Adversary technique matrix for embedded systems, complementing EMB3D's threat model with operational tactics and techniques specific to hardware and firmware environments. Applicable to deployments where TEE-protected workloads operate on or interface with embedded platforms in defense, critical infrastructure, and industrial contexts. Full technique-to-evidence-layer mapping planned for future release.
NIST AI RMF 1.0 / NIST IR 8596 (Cyber AI Profile)	AI workload controls (Section 6) informed by NIST AI risk management guidelines. Framework supports NIST CSF 2.0 mapping via the Compliance Control Mapping annex.
NIST Special Publications (SP) 800-53	Compliance annex provides illustrative mapping of framework evidence to selected 800-53 controls.
CCC Attestation SIG	Post-handshake RA-TLS analysis and relay attack formal proofs inform TB-6 disclosure requirements.
MITRE ATT&CK for ICS	Informative overlay for OT/ICS deployments. Technique mappings (T0859, T0831, T0857, T0886) to framework evidence layers defined in Section 9.4.
MITRE ATT&CK Enterprise (Cloud and Container)	Informative reference for enterprise cloud and container-specific adversary techniques. Complements EMB3D's embedded device focus with cloud-native threat patterns relevant to the framework's orchestration and runtime layers.
MITRE ATLAS (Adversarial Threat Landscape for AI Systems)	Informative reference for adversary techniques targeting AI and machine learning systems. AI workload controls (Section 6) address the subset of ATLAS techniques detectable through infrastructure attestation.

11. Implementation Guidance

This section provides non-normative guidance for organizations implementing this framework.

11.1 Attestation Depth Profiles

The framework supports three deployment profiles corresponding to increasing attestation depth. Implementers select the profile appropriate to their risk tolerance and operational requirements:

Table 21: Attestation Depth Profiles

Profile	Layers Active	Appraiser / Publisher Required	Use Case
Hardware	L1 only	No	Proves the workload runs inside a genuine TEE. Minimum viable confidential computing.
Workload	L1 + L2	Yes	Proves what software is running inside the TEE. Suitable for compliance-driven deployments.
Full	L1 + L2 + L3	Yes	Continuous runtime proof. Required for Zero Trust, cATO, and high-security AI workloads.

The Depth Profile composes orthogonally with the Provider Trust Stance defined in Section 5.3. A complete deployment declaration names both: the Depth Profile (Hardware, Workload, or Full) and the Provider Trust Stance (Provider-Inclusive, Provider-Limited, or Provider-Independent). For example, a deployment may declare "Workload Depth, Provider-Inclusive" for an Infrastructure Consumer operating within a single CSP's trust boundary, or "Full Depth, Provider-Independent" for a regulated workload requiring continuous runtime evidence with silicon-rooted trust anchors only. The two declarations are independent and the implementation must support the combination it claims.

11.2 State-Gated I/O Pattern

A recommended implementation pattern is "State-Gated I/O," where cryptographic keys required by the workload (for decrypting data, accessing storage, etc.) are only released from the KMS after attestation passes. If any layer transitions to FAIL, keys are revoked, and the workload effectively becomes inert.

Trust Recovery. When a Target Machine transitions to FAIL, the framework recommends full redeployment rather than in-place recovery. The replacement

Target Machine must complete a full attestation cycle before receiving trust grants. Outstanding grants from the failed Target Machine are not transferable.

Fleet deployments with multiple Target Machines should implement automated recovery procedures with appropriate human-in-the-loop controls based on failure severity. Recovery policies and their automation boundaries must be documented.

11.3 Compliance Control Mapping

The framework provides a methodology – not a certification – for mapping attestation evidence to individual technical controls within recognized compliance standards (e.g., Cybersecurity Maturity Model Certification (CMMC), Payment Card Industry (PCI) Data Security Standard (DSS), NIST 800-53, Health Insurance Portability and Accountability Act (HIPAA) Security Rule). This mapping is scoped exclusively to machine-verifiable technical controls that can be evidenced through the attestation stack. Organizational and procedural controls (personnel training, incident response planning, access management policies) are explicitly out of scope.

The Appraiser / Publisher can issue individual cryptographically signed compliance control assertions when attestation evidence satisfies the mapped control requirements. Each assertion includes: the control identifier, the evidence elements that satisfy it, a timestamp, the current epoch_id, and a digital signature. These assertions are machine-readable and can be consumed by relying parties for near real-time compliance verification.

Appraiser Policy Attestation: The Appraiser's own configuration – including reference values, enabled checks, and evaluation thresholds – represents a trust-critical input. A misconfigured or tampered Appraiser could sign false PASS verdicts with full authority. Implementations should include the hash of the loaded Appraiser policy configuration as an L2 evidence element, consistent with the treatment of guardrail configuration in AI-L2.4 (Section 6.2).

NOTE: Compliance control mapping is an informative annex to this framework, not a normative requirement. The mapping methodology will be developed in collaboration with compliance assessors and published as a companion document. We included three examples in the ANNEX A.

11.4 Attestation-Gated Workload Identity with SPIFFE / SPIRE

SPIFFE / SPIRE can be used as a complementary mechanism for workload identity in deployments adopting this framework. In this model, the continuous attestation system remains the source of trust regarding platform, image, and runtime integrity, while SPIFFE / SPIRE provides short-lived workload credentials for mutual authentication and authorization between services.

Table 22: Attestation-Gated Workload Identity with SPIFFE/SPIRE

Step	Component	Action	Output
1	Target Machine	Collects L1 / L2 / L3 evidence	Evidence bundle
2	Appraiser	Evaluated evidence against baselines	PASS / ALERT / FAIL verdict
3	Publisher	Signs and streams verdict	Signed attestation event
4	Enterprise Validator	Applies trust policy	Enforcement decision
5	SPIFFE / SPIRE	Issues, renews, or revokes SVIDs based on verdict	SVID credential (on PASS); non-renewal (on FAIL)
6	Services A ⇄ B	Mutual authentication using attestation-gated SVIDs	mTLS session (east-west only)

A logical integration pattern is to bind SPIFFE / SPIRE credential issuance to attestation state:

- A Target Machine in PASS may be authorized to receive or renew an SVID.
- A Target Machine in ALERT may be subject to deployment-specific policy, such as allowing continued credential validity with increased monitoring.
- A Target Machine in FAIL should not receive new SVIDs, and existing credentials should be revoked or allowed to expire without renewal.

This pattern is consistent with the framework's State-Gated I/O guidance. Just as KMS access may be conditioned on attestation state, workload identity issuance may also be conditioned on attestation state. The result is a unified trust enforcement model in which cryptographic identity, key access, and network authorization derive from the same continuous appraisal pipeline.

SPIFFE / SPIRE is most applicable to east-west service authentication within a deployment. It is not, by itself, sufficient to satisfy the End User trust problem addressed by Attested Transport Binding. End User verification requires a cryptographic binding between attestation evidence and the transport session as defined by TB-1 through TB-6. A SPIFFE identity may authenticate a service within a service mesh or internal API fabric, but it does not independently prove that the endpoint is a genuine, currently healthy TEE unless the implementation explicitly binds SVID issuance and acceptance to attestation results.

SVID lifetimes should not materially exceed the attestation interval. A recommended ceiling is two to three times the configured attestation interval. Longer credential lifetimes create a window during which a workload may retain a valid identity credential

after its attestation state has degraded. Deployments that require longer SVID lifetimes for operational reasons should document the resulting exposure window and compensate with additional monitoring or faster revocation paths.

11.5 Cryptographic Key and Signature Overview

The framework references several categories of cryptographic keys used for signing, verification, and encryption at different layers. The following table provides an overview of each key category, its owner, protection model, and the role responsible for verification.

Table 23: Cryptographic Key and Signature Overview

Key Category	Owner	Protection Model	Verified By
Hardware attestation signing key (e.g., AMD VCEK, Intel QE)	Silicon vendor (AMD, Intel, Arm)	Fused in hardware; not extractable. Certificate chains published by vendor.	Appraiser (L1); End User (via TB-1 and cert chain)
GPU attestation signing key	GPU vendor (e.g., NVIDIA)	GPU hardware Root of Trust; independent of CPU attestation key.	Appraiser (L1-GPU via vendor JWKS or cert chain)
CI/CD image signing key (e.g., Sigstore/Cosign)	Software vendor or Tenant	HSM-backed or KMS-backed; managed through software supply chain.	Appraiser (L2 baseline verification)
Composite Baseline vendor signing key	Platform vendor	Managed by platform vendor; signs immutable vendor baselines (Section 4.2.2).	Appraiser (baseline discrimination mechanism)
Publisher event signing key (e.g., ML-DSA per FIPS 204)	Generated inside TEE at Publisher startup	TEE memory isolation; public key hash embedded in attestation evidence.	Enterprise Validator (event signature verification)
TLS session private key	Generated inside TEE	Varies by implementation (see TB-5). Ranges from guest memory to Secure VM Service Module (SVSM).	End User (via TLS handshake + TB-1 key commitment)
KMS encryption keys	Cloud KMS provider	KMS infrastructure; released only on PASS verdict (Section 11.2).	Workload (decryption); KMS (access policy enforcement)

NOTE: This table is informative and is intended to assist implementers in understanding the trust relationships between keys. Not all key categories are present in all deployments; the set of active keys depends on the deployment profile (Section 11.1) and the specific TEE platform.

11.6 Cryptographic Algorithm Requirements and Post-Quantum Readiness

Cryptographic algorithms referenced by this framework fall into three categories with distinct post-quantum migration characteristics.

Framework-controlled cryptography includes Publisher event signing keys, compliance assertion signatures, and key encapsulation for state-gated I/O. These algorithms are selected by the implementation and have no external dependency. Implementations must use a NIST-standardized post-quantum signature algorithm (e.g., ML-DSA per FIPS 204) for event and assertion signing, and a NIST-standardized post-quantum key encapsulation mechanism (e.g., ML-KEM per FIPS 203) for key transport operations.

Platform-dependent cryptography includes TLS key exchange and transport binding mechanisms. Post-quantum key exchange is available (e.g., ML-KEM hybrid key exchange such as SecP256r1MLKEM768) and should be deployed where supported by the TLS implementation and intermediary infrastructure.

Hardware-locked cryptography includes attestation report signatures produced by the silicon vendor's Root of Trust (e.g., AMD VCEK, Intel QE, NVIDIA GPU RoT). These signatures use algorithms determined by the hardware vendor and cannot be changed by the framework or the implementation. As silicon vendors migrate their attestation signing to post-quantum algorithms, deployments will benefit automatically without requiring a framework revision.

Symmetric algorithms used by the framework (AES-256-GCM) are already considered quantum-safe and require no migration.

Implementations must disclose the cryptographic algorithms in use at each layer as part of the attestation metadata, including which algorithm categories remain classical due to hardware dependencies. This disclosure follows the same transparency model used for L3.1 coverage limitations (Section 4.3.1) and key protection disclosure (TB-5) and enables Enterprise Validators and End Users to evaluate the deployment's overall cryptographic posture.

12. Future Work

- **CoRIM Migration:** Migrate Golden Baseline format from YAML Ain't Markup Language (YAML) to IETF CoRIM (Concise Reference Integrity Manifest) for standards-aligned reference value distribution.

- **AR4SI Alignment:** Align Appraisal Result format with IETF Attestation Results for Secure Interactions (AR4SI) as the specification matures.

- **GPU TEE Attestation Normative Requirements:** Elevate the informative GPU TEE guidance in Section 4.1.2 to normative requirements following vendor engagement and community feedback on this draft.

- **L7 Transport Binding Solutions:** Investigate and standardize approaches for maintaining transport binding through L7 load balancers and CDN edge nodes (see Section 8.5).

- **Open-Source Reference Implementation:** Develop an open-source reference implementation of the framework to enable community validation and accelerate adoption.

- **Agent-to-Agent Attestation:** Define attestation requirements for agentic AI systems where multiple TEE-hosted agents communicate and make autonomous decisions, including mid-session re-attestation protocols and agent identity registries.

- **Distributed Training Attestation:** Define composite attestation requirements for multi-node training deployments where each GPU node operates as an independent TEE, including cross-node attestation coordination and aggregate verdict computation.

- **GPU Behavioral Evidence Collection:** Collaborate with GPU vendors to define standardized telemetry interfaces for GPU-specific behavioral evidence (memory access patterns, compute utilization profiles) suitable for continuous attestation.

- **Cryptographic Trust Chain Companion Document:** Develop a detailed companion document expanding the key and signature overview (Section 11.4) with per-platform key lifecycle details, certificate chain validation procedures, and cross-platform key management guidance.

- **Axioms for Cyber Integrity:** Investigate formalizing the framework's trust assumptions as a set of explicit axioms, drawing on established work in cyber integrity reasoning. Such axioms could provide a structured basis for threat analysis (e.g., "this threat materializes when Axiom X is not achieved") and facilitate formal verification of attestation protocols.

- **Architectural Diagrams:** Develop a consistent set of architectural diagrams illustrating the framework's roles, evidence flow, and integration patterns (including SPIFFE / SPIRE attestation-gated identity) as a companion to the textual specification.

- **MITRE ATLAS Mapping:** Extend the AI workload controls (Section 6) with a systematic mapping to MITRE ATLAS adversary techniques, identifying which techniques are mitigable through infrastructure attestation and which require application-layer defenses.

- **ESTM Technique Mapping:** Develop a systematic mapping of MITRE ESTM (Embedded Systems Threat Matrix) techniques to the framework's evidence layers, extending the threat model coverage for defense and critical infrastructure deployments where TEE-protected workloads operate on embedded platforms. This mapping would complement the existing ATT&CK for ICS overlay (Section 9.4) and provide a structured basis for embedded system attestation requirements.

- **Post-Quantum Cryptography (PQC) Migration Guide:** Develop a companion document detailing the full PQC transition path for continuous attestation deployments, including hardware attestation signing migration timelines, intermediate cryptographic postures during transition, cryptographic agility requirements, and CNSA 2.0 alignment guidance.

13. Glossary

Table 24: Glossary

Term	Definition
Appraiser	The role that compares measured evidence against reference baselines and produces a trust verdict.
ATT&CK for ICS	MITRE's threat knowledge base for Industrial Control Systems, cataloging adversary tactics and techniques observed in attacks against OT/ICS environments. Used as an informative overlay to the framework's threat model for OT deployments (Section 9.4).
Attested Transport Binding (ATB)	The cryptographic property that links a TEE's hardware attestation evidence to a specific TLS session, ensuring the attested environment and the TLS endpoint are the same entity. Defined by requirements TB-1 through TB-6 in Section 8.
aTLS / RA-TLS	Remotely Attested TLS. A class of protocols that implement Attested Transport Binding by embedding a cryptographic commitment to the TLS session key inside TEE attestation evidence. RA-TLS is one approach to satisfying the transport binding requirements defined in Section 8.
Attestation Metadata	Structured information published alongside attestation evidence and appraisal results. Includes the active layer set, attestation interval, evidence gathering technique and coverage limitations, deployment profile, and other parameters necessary for relying parties to interpret results. Aligns with metadata concepts in IETF RFC 9334 and is extended in this framework for continuous attestation.
Composite Baseline	A two-tier reference value set combining immutable vendor-defined values with tenant-configurable values. Vendor baselines (kernel, OS, monitoring agents) cannot be overridden by tenant baselines. See Section 4.2.2.
Dead Man's Switch	Automatic trust revocation triggered when a Target Machine stops reporting evidence within a configured window.
eat_nonce	A nonce field in the Entity Attestation Token used to bind attestation freshness to a specific TLS certificate or session.
eBPF	Extended Berkeley Packet Filter. In-kernel programmable hooks used for syscall monitoring, network filtering, and behavioral profiling.
Enterprise Validator	The organization's relying party that consumes published attestation results and enforces trust policy.
Evidence Bundle	A structured collection of evidence from all active layers, transmitted from the Measurement Gatherer to the Appraiser.
Golden Baseline	The expected-state reference values against which measured evidence is compared during appraisal. In practice, the Golden Baseline is constructed as a Composite Reference Baseline (Section 4.2.2) combining vendor and tenant values.
GPU TEE	A GPU-based Trusted Execution Environment providing hardware-enforced memory encryption and attestation for accelerator workloads (e.g., NVIDIA CC).

Term	Definition
ICS (Industrial Control Systems)	Systems used to monitor and control industrial processes, including SCADA (Supervisory Control and Data Acquisition), DCS (Distributed Control Systems), PLCs (Programmable Logic Controllers), and RTUs (Remote Terminal Units).
IMA	Integrity Measurement Architecture. A Linux kernel feature that measures (hashes) executables and libraries at load time.
KIM	Kernel Integrity Monitor. A service that validates kernel data structure integrity at runtime.
Measurement Gatherer	The role that collects raw evidence from the Target Machine at each attestation layer.
OT (Operational Technology)	Hardware and software that detects or causes changes through direct monitoring and control of physical devices, processes, and events. Distinguished from IT by its direct interface with the physical world and its deterministic timing and safety requirements.
Publisher	The role that streams signed appraisal results to Enterprise Validators.
Provider Trust Stance	The policy choice declared by a relying party regarding which trust anchor types it accepts as evidence. The framework defines three stances: Provider-Inclusive (accepts Type A, B, and C), Provider-Limited (selective), and Provider-Independent (accepts only Type A). See Section 5.3.
SBOM	Software Bill of Materials. A machine-readable inventory of software components and their dependencies.
SPIFFE	Secure Production Identity Framework for Everyone. A framework for assigning interoperable cryptographic identities to workloads.
SPIRE	SPIFFE Runtime Environment. A reference implementation of SPIFFE used to issue SPIFFE Verifiable Identity Documents (SVIDs) to workloads based on registration and policy.
Split Verifier	Architectural pattern where evidence gathering and appraisal are separated, with appraisal occurring inside the trust boundary.
SVSM	Secure VM Service Module. An example of hardware-isolated key storage running at a higher privilege level within the TEE, providing vTPM and key isolation services.
Tenant	An organization consuming infrastructure from a provider above it in the deployment stack. An IaaS Tenant (Perspective 2, Section 5) consumes compute from the CSP. A PaaS / SaaS operator may in turn be a tenant of the IaaS provider. Where precision is required, this framework uses the perspective-specific terms defined in Section 5.
Target Machine	The TEE-protected environment where the workload runs.
Trust Anchor Type	The composition of the certificate chain or signed token path that delivers attestation evidence to a relying party. The framework defines three types: Type A (silicon-rooted), Type B (provider-mediated, e.g., a CSP-operated verification service), and Type C (vendor-service-mediated, e.g., a hardware-vendor-operated attestation service). See Section 4.1.3.
TCB	Trusted Computing Base. The set of hardware, firmware, and software that must be trusted for the security of the system.

Term	Definition
TEE	Trusted Execution Environment. Hardware-enforced isolated memory region.
YANG Push	RFC 8641. A protocol for streaming configuration and state change notifications, used here for attestation result publication.

14. References

[1] H. Birkholz, D. Thaler, M. Richardson, N. Smith, W. Pan, “Remote ATtestation procedureS (RATS) Architecture,” RFC 9334, IETF, January 2023.

[2] A. Clemm, E. Voit, “Subscription to YANG Notifications for Datastore Updates,” RFC 8641, IETF, September 2019.

[3] CCC Attestation SIG, “ProVerif Analysis of Intra-Handshake TEE Attestation Protocols,” Confidential Computing Consortium, 2025.

[4] MITRE, “EMB3D: Embedded Device Threat Model,” The MITRE Corporation, 2024.

[5] NIST, “AI Risk Management Framework (AI RMF 1.0),” NIST AI 100-1, January 2023.

[6] NIST, “Cybersecurity Framework Profile for Artificial Intelligence (Cyber AI Profile),” NIST IR 8596 (Preliminary Draft), December 2025.

[7] W. Thomas, L. Schmalz, A. Petz, P. Alexander, P. D. Rowe, J. Guttman, J. Carter, “Designing Trustworthy Layered Attestations,” 2026.

[8] Sigstore Project, “Rekor: Transparency Log for Software Supply Chain,” Linux Foundation, 2023.

[9] NIST, “Security and Privacy Controls for Information Systems and Organizations,” NIST SP 800-53 Rev. 5, September 2020.

[10] NIST, “Launching the AI Agent Standards Initiative,” NIST CAISI, January 2026.

[11] P. D. Rowe, J. Ramsdell, I. Kretz, “Automated Trust Analysis of Copland Specifications for Layered Attestations,” Principles and Practice of Declarative Programming (PPDP), September 2021.

[12] P. D. Rowe, “Confining Adversary Actions via Measurement,” Third International Workshop on Graphical Models for Security, 2016.

[13] MITRE, "ATT&CK for Industrial Control Systems", The MITRE Corporation.

15. Abbreviations and Acronyms

Term	Definition
AI	Artificial Intelligence
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
AMD	Advanced Micro Devices
API	Application Programming Interface
AR4SI	Attestation Results for Secure Interactions (IETF)
ARK	AMD Root Key
ATLAS	Adversarial Threat Landscape for AI Systems
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge (MITRE)
AWS	Amazon Web Services
CA	Certificate Authority
CAISI	Center for AI Standards and Innovation (NIST)
CC	Confidential Computing
CCA	Confidential Compute Architecture (Arm)
CCC	Confidential Computing Consortium
CDN	Content Delivery Network
CIP	Critical Infrastructure Protection
CMMC	Cybersecurity Maturity Model Certification
CNSA	Commercial National Security Algorithm Suite
CNCF	Cloud Native Computing Foundation
CPU	Central Processing Unit
CSP	Cloud Service Provider
DKOM	Direct Kernel Object Manipulation
DNS	Domain Name System
DSS	Data Security Standard (as in PCI DSS)
EAT	Entity Attestation Token
ECDH	Elliptic Curve Diffie-Hellman
EMB3D	MITRE EMB3D (Embedded Threat Model framework)
ESTM	Embedded Systems Threat Matrix (MITRE)
FIPS	Federal Information Processing Standards
GB	Gigabyte

Term	Definition
GCM	Galois/Counter Mode
GCP	Google Cloud Platform
GPU	Graphics Processing Unit
HIPAA	Health Insurance Portability and Accountability Act
HTTP	Hypertext Transfer Protocol
HTTP/2	Hypertext Transfer Protocol Version 2
ICS	Industrial Control Systems
ID	Identifier
IDT	Interrupt Descriptor Table
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IMA	Integrity Measurement Architecture (Linux)
IR	Interagency Report
IT	Information Technology
IT/OT	Information Technology / Operational Technology
JIT	Just-In-Time
JSON	JavaScript Object Notation
JWT	JSON Web Token
KEM	Key Encapsulation Mechanism
KMS	Key Management Service
L1–L7	Layers 1–7 of the OSI networking model
MITRE	MITRE Corporation
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
NERC	North American Electric Reliability Corporation
NETCONF	Network Configuration Protocol
NIST	National Institute of Standards and Technology
NRAS	NVIDIA Remote Attestation Service (referenced as attestation token source)
NVD	National Vulnerability Database
OS	Operating System
OSV	Open Source Vulnerabilities database
OT	Operational Technology

Term	Definition
PCK	Provisioning Certification Key (Intel)
PCI DSS	Payment Card Industry Data Security Standard
PKI	Public Key Infrastructure
PPDP	Principles and Practice of Declarative Programming
PQC	Post-Quantum Cryptography
PSK	Pre-Shared Key
PSP	Platform Security Processor (AMD)
QE	Quoting Enclave (Intel SGX)
RA	Remote Attestation
RAG	Retrieval-Augmented Generation
RATS	Remote ATtestation procedureS (IETF architecture)
RESTCONF	RESTful Network Configuration Protocol
RFC	Request for Comments
RMF	Risk Management Framework
RTT	Round Trip Time
SBOM	Software Bill of Materials
SEV	Secure Encrypted Virtualization (AMD)
SGX	Software Guard Extensions (Intel)
SIG	Signature / Special Interest Group (context-dependent)
SNP	Secure Nested Paging (AMD SEV-SNP)
SPIFFE	Secure Production Identity Framework for Everyone
SPIRE	SPIFFE Runtime Environment
SP	Special Publication (e.g., NIST SP 800-series)
SSE	Server-Sent Events
SVID	SPIFFE Verifiable Identity Document
TCB	Trusted Computing Base
TCP	Transmission Control Protocol
TDX	Trust Domain Extensions (Intel)
TEE	Trusted Execution Environment
TLS	Transport Layer Security
TPM	Trusted Platform Module
TTL	Time To Live

Term	Definition
UDP	User Datagram Protocol
URL	Uniform Resource Locator
VCEK	Versioned Chip Endorsement Key (AMD)
VM	Virtual Machine
VMPL	Virtual Machine Privilege Level (AMD SEV-SNP)
WAF	Web Application Firewall
YAML	YAML Ain't Markup Language
YANG	Yet Another Next Generation (data modeling language)