Securing AI-Enabled Critical Systems: From Software-Level Protection to Hardware–Software Co-Design

Technical White Paper

Maryam Zarezadeh
Barkhausen Institut
Dresden, Germany
maryam.zarezadeh@barkhauseninstitut.org

Gerhard P. Fettweis
Barkhausen Institut, Dresden, Germany
Vodafone Chair Mobile Communications Systems,
Technische Universität Dresden, Dresden, Germany
gerhard.fettweis@barkhauseninstitut.org

***Abstract*—In critical systems, even a correctly executed AI model can recommend an unsafe action. This white paper argues that the authority granted to an AI component and the fallback behavior required when it fails or becomes unavailable should guide the choice of protection mechanisms. Using industrial monitoring as a running example, the paper examines how to protect the AI lifecycle from data collection and model development to execution and operational use. It distinguishes computational integrity, which concerns whether the approved computation was performed, from operational validity, which concerns whether the proposed action is appropriate for the current state of the physical process. The proposed hardware–software architecture combines controlled model development, authenticated lifecycle records, trusted execution or accelerator verification, and independent authorization before consequential actions. It addresses stale data, shared dependencies such as time synchronization, and attacks that deliberately force degraded operation. The paper relates these controls to IEC 62443, functional safety standards, NIST guidance, and the EU AI Act. It presents a taxonomy linking lifecycle design choices to hardware trust boundaries, execution evidence, and operational constraints. The paper identifies research priorities in protected inference, numerical precision, resistance to leakage and faults, and recovery, and outlines a staged roadmap for validating the proposed architecture.**

***Index Terms*—AI security, critical systems, operational technology, computational integrity, operational validity, trusted execution, hardware–software co-design, resilience, IEC 62443, functional safety.**

1. Introduction

Consider an industrial AI monitor that flags unusual network traffic and recommends isolating a controller. The approved model may have run correctly, but it may have relied on stale telemetry. Acting on its recommendation could interrupt a process that still depends on the controller. In critical systems, compromise or failure can disrupt essential services or contribute to physical harm. Incidents in operational technology show that this is not hypothetical: Stuxnet manipulated industrial controllers while presenting normal values to operators [1], the 2015 attack on the Ukrainian power grid combined intrusion into control networks with disruption of the operators' ability to respond [2], and TRITON targeted a safety instrumented system, the last line of protection in a process plant [3]. Each case attacked the link between what operators observe and what the process does. An AI component inserted into that link becomes both a new defensive capability and a new target.

Protection must therefore extend beyond the model to the complete path from collection to the operational consequences of a decision. This path crosses software, execution hardware, verification services, and process safeguards. Optimizing or securing these components separately can leave gaps at their interfaces: a protected computation can consume stale observations, an authenticated model can run on an unverified accelerator, and a valid result can still recommend an unsafe intervention. Sculley et al. identify data dependencies, hidden feedback loops, and configuration issues as sources of system-level technical debt in machine learning, motivating attention to the surrounding system throughout its lifecycle [4].

Figure 1 previews the central design logic: authority and fallback are fixed before protection mechanisms are selected, software policies are connected to execution evidence, and the resulting recommendation must pass an independent operational check. The diagram describes the proposed assurance path, not an implemented system.

1.1. Purpose and intended use

This white paper is intended to guide early design decisions for organizations considering AI in the monitoring or operation of critical systems, and to set out a research agenda for hardware–software co-design in this setting. Its readers are cybersecurity leaders, operational technology engineers, AI researchers, and hardware designers. It synthesizes existing guidance and research into a cross-layer security argument, identifies design decisions that couple software controls to hardware enforcement and operational constraints, and proposes an architecture and evaluation

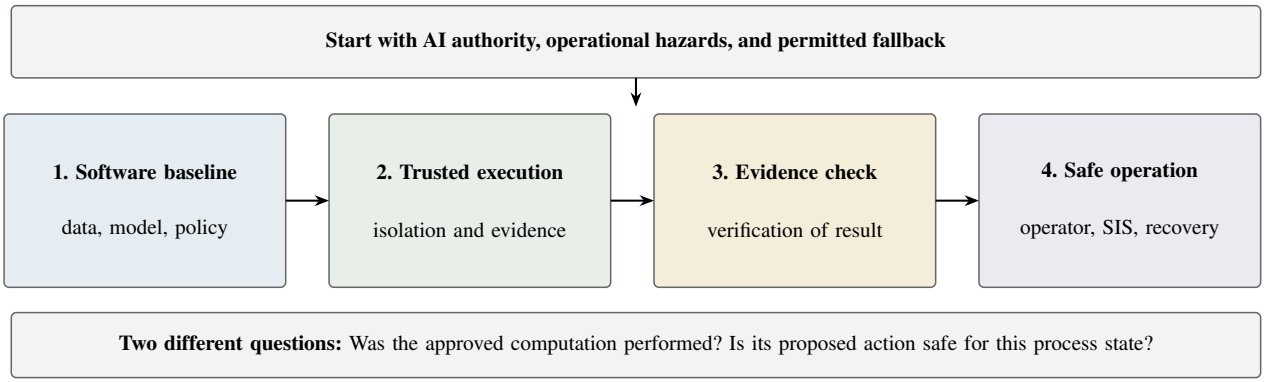


Figure 1. Security assurance at a glance. Software controls, trusted execution, and independent authorization form a connected path; none independently establishes that an AI recommendation is safe to act upon. The design starts with a defined authority boundary and fallback procedure. Blue denotes software/data handling, green protection and enforced execution, amber evidence verification, and violet action authorization; neutral elements cover operational safety and other roles. Threats and security incidents are red where shown. Labels and line styles retain meaning in grayscale.

plan; it does not report a validated deployment, and sector-specific hazard analysis and assurance remain necessary. A second case, distributed radio-frequency (RF) sensing, examines how to divide inference between antenna nodes and a central accelerator while protecting data and verifying the contributions used to produce each result.

1.2. Contributions

The paper makes the following contributions:

- It distinguishes *computational integrity* from *operational validity* and argues that a critical-system decision requires evidence for both, obtained through different mechanisms (Section 6.3).
- It relates the protection an AI component requires to its granted authority and explains why confidentiality requirements depend on asset sensitivity rather than authority level (Section 3.4, Fig. 2).
- It maps threats and controls across the AI lifecycle, including dataset admission and shared dependencies such as time synchronization that can undermine apparently independent safeguards (Section 4).
- It proposes an architecture with explicit trust boundaries and a degraded-operation design that addresses attacks intended to make AI assistance unavailable (Section 7).
- It develops a cross-layer taxonomy of design decisions and trade-offs linking software controls, execution hardware, verification, and operational safeguards, and outlines validation requirements, staged adoption, and go/no-go criteria (Sections 8 and 9).

The paper is organized as follows. Section 2 summarizes the standards and regulation that frame the problem. Section 3 defines AI roles and authority, and Section 4 the threat model. Sections 5 and 6 examine software and hardware protection. Section 7 presents the architecture, Section 8 the co-design priorities, and Section 9 outlines validation and adoption considerations. Section 10 discusses limitations and Section 11 concludes.

2. Standards and regulatory landscape

No single standard covers AI security in critical systems. The relevant guidance comes from three communities that have developed largely independently: industrial cybersecurity, functional safety, and AI governance. Table 1 summarizes the instruments used in this paper and where each is applied.

2.1. Industrial cybersecurity

The IEC 62443 series is the principal international framework for the security of industrial automation and control systems. IEC 62443-3-2 structures a system into *zones* of assets with common security requirements, connected by *conduits* that carry and control communication between them, and assigns a target security level to each zone and conduit [5]. IEC 62443-3-3 defines system security requirements and four security levels that correspond to increasingly capable adversaries, from casual violation to sophisticated attacks with extended resources and system-specific knowledge [6]. IEC 62443-4-1 specifies a secure product development lifecycle [7]. This paper treats the AI service as a component that must be placed in a defined zone with explicit conduits (Section 5.4), and uses the security-level concept to state adversary capability in the threat model (Section 4.3).

NIST SP 800-82 Rev. 3 addresses OT security in the context of performance, reliability, and safety requirements [8]. This paper uses that final revision as its baseline. An initial public draft of Revision 4, aligned with the NIST Cybersecurity Framework 2.0, was released in September 2026 for comment [9]; readers should check for the final version.

2.2. Functional safety

IEC 61508 is the generic functional safety standard for electrical, electronic, and programmable electronic safety-related systems [10], and IEC 61511 applies it to safety

instrumented systems in the process industry [11]. These standards define safety integrity levels and require that a safety function be designed and verified against specified failure rates. Machine learning components do not fit easily into these frameworks because their failure behavior is not characterized by the methods the standards assume. This paper therefore does not place AI inside a safety function; it requires that safety functions remain independent of the AI path (Section 7.3). IEC TR 63069 provides a framework for the coexistence of functional safety and security, including the principle that security measures must not compromise safety functions [12]. Systems-theoretic hazard analysis such as STPA is useful for identifying unsafe control actions that an AI recommendation could trigger [13].

2.3. AI governance and threat knowledge

NIST's AI Risk Management Framework treats safety, security, resilience, and validity as related aspects of trustworthy AI [14]. NIST's preliminary Cyber AI Profile applies the Cybersecurity Framework to three focus areas: securing AI components, AI-enabled cyber defense, and thwarting AI-enabled attacks [15]. That publication remained a preliminary draft at the time of writing. ISO/IEC 42001 specifies requirements for an AI management system [16]. For threat terminology, this paper uses the NIST adversarial machine learning taxonomy [17], MITRE ATLAS [18], and, for language-model applications, the OWASP Top 10 for LLM Applications [19]. Joint guidance from national cybersecurity agencies covers secure AI development [20], secure deployment [21], and the integration of AI into OT [22].

2.4. European regulation

For organizations in the European Union, three instruments are directly relevant. The AI Act classifies as high-risk those AI systems intended to be used as safety components in the management and operation of critical digital infrastructure, road traffic, or the supply of water, gas, heating, or electricity (Annex III, point 2) [23]. High-risk systems must meet requirements including human oversight that addresses automation bias (Article 14) and an appropriate level of accuracy, robustness, and cybersecurity, with measures against data poisoning, model poisoning, adversarial examples, and confidentiality attacks (Article 15). Under the amended Article 113, Sections 1–3 of Chapter III, except Article 6(5), apply from 2 December 2027 to AI systems classified as high-risk under Article 6(2) and Annex III [24]. Whether a given system falls in scope depends on its intended purpose: an advisory detector with no safety role may fall outside Annex III, whereas the same model granted authority over a safety-relevant action is more likely to fall within it. The authority levels in Section 3.4 are therefore also a regulatory question.

The NIS2 Directive requires essential and important entities in sectors such as energy, transport, and water to implement cybersecurity risk-management measures, including supply-chain security, and to report significant incidents [25]. The Cyber Resilience Act sets cybersecurity requirements for products with digital elements, including vulnerability handling and security updates over the product's support period [26]; it is relevant to vendors of AI-enabled OT products and to the update mechanisms discussed in Section 8.4.

3. AI roles and operational consequences

Two related problems must be addressed. AI for cybersecurity includes detecting anomalies, correlating alerts, and helping analysts investigate incidents. Cybersecurity for AI includes protecting datasets, model artifacts, infrastructure, and decision interfaces from compromise. Both matter when an AI security tool can influence a critical service, because a compromised defensive tool can blind operators or trigger harmful responses.

3.1. Criticality depends on the consequence of failure

A system is critical here when its disruption or incorrect behavior can materially affect essential services, safety, or mission continuity. Operational technology, or OT, monitors or changes physical processes. Unlike most IT systems, OT often prioritizes availability and deterministic timing over confidentiality, runs equipment for decades, and cannot always be patched or restarted on demand [8]. These properties constrain which security mechanisms can be deployed and how failures may be handled.

Table 2 summarizes the proposed authority boundaries.

3.2. Running example

In the industrial monitoring example introduced in Section 1, the model analyzes network telemetry and equipment measurements. The model estimates whether activity is abnormal and presents evidence to an operator. A compromised model could miss an intrusion or falsely recommend isolating a controller. An attacker could also manipulate telemetry so that a correctly executed model reaches a misleading conclusion. The proposed initial deployment therefore has no direct actuator or firewall-change privileges. The example is used throughout the paper; where a mechanism is not needed at the advisory level, this is stated explicitly.

3.3. Security and safety require distinct arguments

Confidentiality limits disclosure, integrity limits unauthorized modification, and availability supports continuity. Safety concerns unacceptable harm, including harm caused without an attacker. Resilience concerns continued or recovered operation under disruption. A security test does not replace a hazard analysis, and strong detection accuracy does not establish safe automation.

For every use case, name the affected assets, the permitted action, and the consequence of a false positive, a

TABLE 1. STANDARDS, GUIDANCE, AND REGULATION USED IN THIS PAPER

Instrument	Scope relevant here	Where applied
IEC 62443-3-2, -3-3, -4-1 [5]–[7]	Zones and conduits; security levels; secure development lifecycle	Network placement (Sec. 5.4); adversary capability (Sec. 4.3); build controls (Sec. 5.3)
IEC 61508, IEC 61511, IEC TR 63069 [10]–[12]	Functional safety; safety instrumented systems; safety–security coexistence	Independence of protection (Sec. 7.3); fallback design (Sec. 7.2)
NIST SP 800-82 Rev. 3 / Rev. 4 draft [8], [9]	OT security architecture and practice	Baseline OT controls throughout
NIST AI RMF, Cyber AI Profile, AML taxonomy [14], [15], [17]	Trustworthy-AI characteristics; CSF profile for AI; attack terminology	AI roles (Sec. 3); threat model (Sec. 4)
MITRE ATLAS, OWASP LLM Top 10 [18], [19]	Adversary techniques against AI systems	Threat model and test cases (Secs. 4, 9)
EU AI Act as amended [23], [24]	High-risk classification; human oversight; robustness and cybersecurity	Authority levels (Sec. 3.4); oversight (Sec. 7.3)
NIS2, Cyber Resilience Act [25], [26]	Operator risk management and incident reporting; product security and updates	Supply chain (Sec. 5.3); updates and recovery (Sec. 8.4)
RFC 9334, NIST SP 800-193 [27], [28]	Remote attestation architecture; firmware resilience	Attestation (Sec. 6.1); recovery (Sec. 8.4)

TABLE 2. AI AUTHORITY LEVELS AND OPERATIONAL BOUNDARIES

AI role	Illustrative use	Proposed authority boundary
Advisory detection	Flag unusual industrial network traffic	Read-only analysis; analyst reviews alerts
Decision support	Recommend maintenance or incident containment	Named operator authorizes consequential changes
Bounded automation	Execute a pre-approved limited response	External rules constrain assets, timing, and actions
Operational control	Influence a physical process directly	Separate sector-specific safety case and runtime constraints

missed event, or a late result. These decisions determine the assurance evidence needed before expanding the system's authority.

3.4. Matching protection to authority

Protection should scale with the authority granted to the AI component. Fig. 2 proposes an illustrative minimum for each level of Table 2. At the advisory level, the dominant risks are a missed event and a misleading alert; both are mitigated by a secure software baseline and by an operator who can compare the alert with other evidence. Once the AI's output can trigger an action without case-by-case human review, the relying party can no longer catch a tampered result, and verified execution becomes necessary. At the operational-control level, a sector-specific safety case is required in addition.

Confidentiality does not follow this staircase. Model weights may need protection even at the advisory level, for two reasons. First, a detector's weights can encode sensitive process knowledge. Second, an adversary who obtains the model, or a functionally equivalent copy through repeated queries [29], can craft manipulated telemetry offline that evades it. Erba et al. demonstrate such concealment attacks against reconstruction-based anomaly detectors in industrial control systems [30]. Confidential execution is therefore justified by the sensitivity of the assets and by the evasion risk, independently of authority.

4. Threat model across the AI lifecycle

The proposed threat model follows information and control across organizational boundaries. Assets include sensor measurements, network logs, training labels, model weights, credentials, configuration files, and operational commands. The adversary may control an external data source, a service account, a host operating system, or a delegated accelerator. These capabilities must be stated separately. The NIST adversarial machine learning taxonomy [17] and MITRE ATLAS [18] provide terminology for poisoning, evasion, extraction, and privacy-related threats; Biggio and Roli review the development of these attacks [31].

Table 3 summarizes the lifecycle threat model, and Fig. 3 pairs each stage with representative controls from Sections 5 and 6.

4.1. Evasion of process anomaly detectors

An anomaly detector for a physical process faces a constrained adversary: manipulated values must remain physically plausible enough not to trip conventional alarms, and the attacker may control only some sensors. These constraints do not make evasion impractical. Concealment

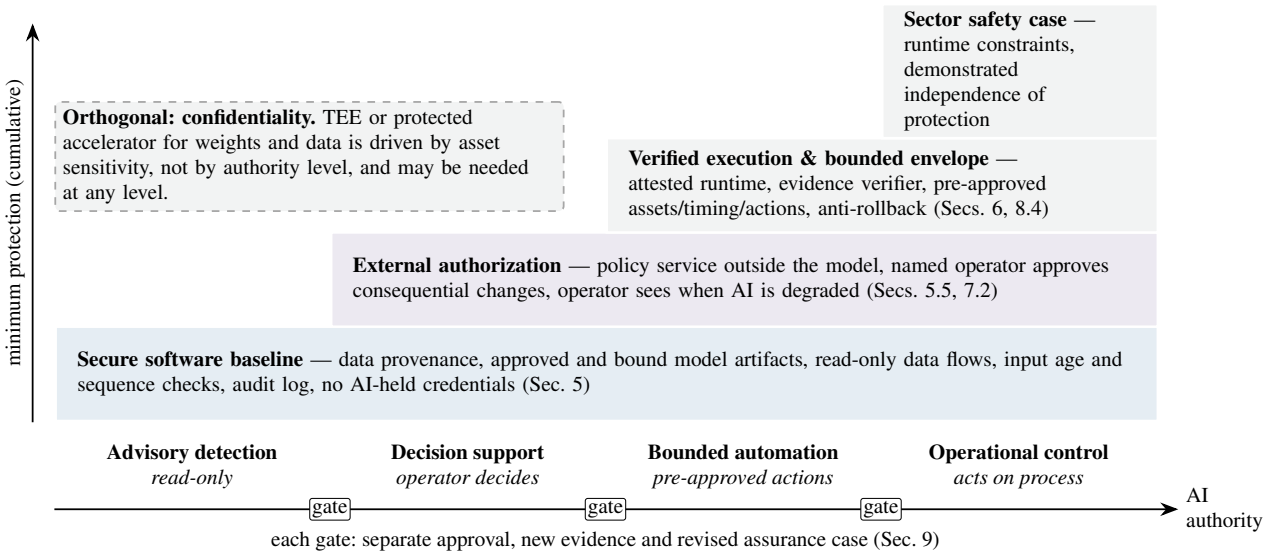


Figure 2. Authority-dependent protection requirements (proposed). Controls are cumulative; moving to greater AI authority requires separate approval and a revised assurance case. Confidentiality depends on asset sensitivity, and sector-specific hazards determine the actual requirements.

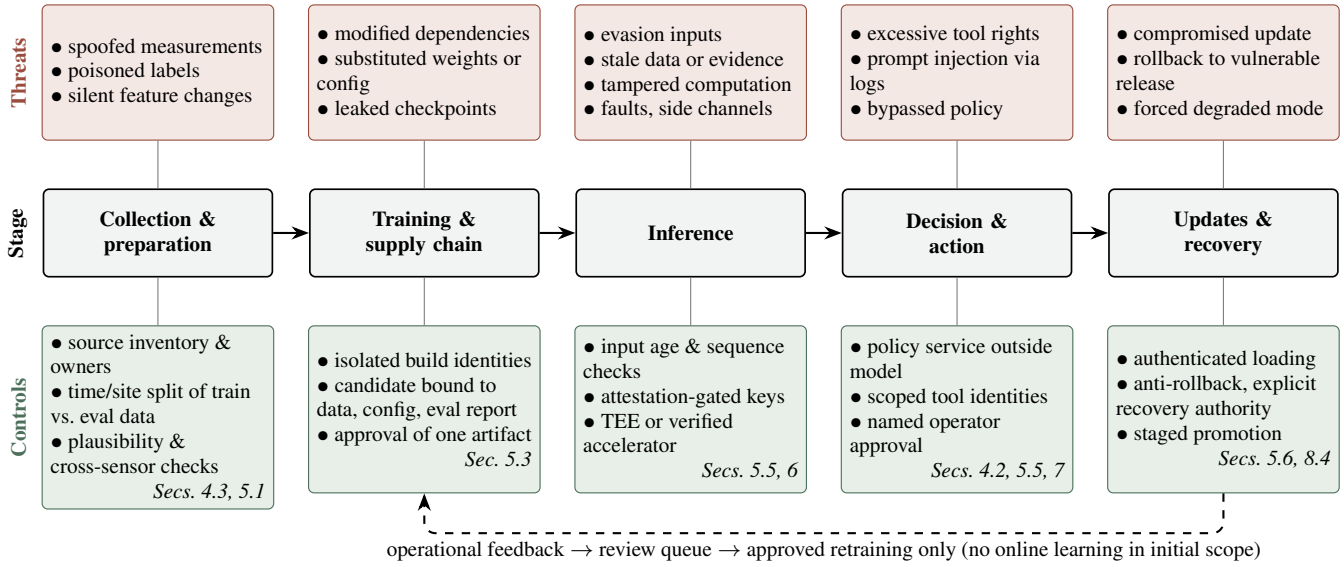


Figure 3. Threats and corresponding controls across the AI lifecycle. Operational feedback enters training only after review; online learning is outside the proposed initial scope.

attacks can hide process anomalies from reconstruction-based detectors while modifying a limited subset of sensor readings [30], which is the attack pattern Stuxnet applied against human operators [1]. The threat model should therefore state which sensors and network segments the attacker can modify, and evaluation should include adaptive attacks that know the detector, not only attacks drawn from a fixed dataset (Section 9).

4.2. Generative AI and security assistants

If a language model summarizes logs or uses operational tools, attacker-controlled text is another input channel. Indirect prompt injection, in which instructions are embedded in content the model retrieves rather than in the user's request, has been demonstrated against deployed LLM-integrated applications [32] and is the leading risk in the OWASP Top 10 for LLM Applications [19]. In a security operations setting, an attacker who can write to a log file, ticket, or firmware banner can place such instructions in front of the assistant. Treat retrieved documents, tickets, and log

TABLE 3. THREATS AND CONSEQUENCES ACROSS THE AI LIFECYCLE

Stage	Representative threat	Potential consequence
Collection and preparation	Spoofed measurements or poisoned labels	The model learns or observes a false operating picture
Training and supply chain	Modified dependencies, weights, or configuration	Unauthorized behavior enters the released system
Inference	Evasion, stale inputs, tampered computation, or faults	Missed incident or misleading recommendation
Decision and action	Excessive tool permissions or bypassed policy	An AI error becomes an operational change
Updates and recovery	Compromised update, rollback, or forced degraded mode	Protection degrades after initial approval

messages as untrusted content, not instructions that can grant permissions.

The proposed integration uses narrowly scoped tool identities, structured arguments validated against a schema, and authorization outside the model. Tools that can change firewall rules, accounts, or equipment configuration should require the same policy check and operator approval as any other consequential action (Section 5.5). A fluent explanation should never be accepted as proof that an action is authorized, and the assistant's output should cite the underlying records so that the operator can verify them.

4.3. Assumptions and exclusions

The first prototype should trust a small approval service, its key-management process, and the selected hardware root of trust. It explicitly models compromised host software and modified stored artifacts. Sensor compromise, physical faults, side-channel observation, and denial of service have separate test cases. Invasive hardware attacks and malicious administrators of the approval service are excluded from the first prototype; this exclusion is part of the security claim. Expressed in IEC 62443 terms, the prototype targets an adversary with moderate resources and system-specific knowledge (security level 3) [6] for the analytics zone; a higher target would require protection of the approval service against its own administrators.

A compromised source may sign false measurements with a valid key. Authentication identifies a source; it does not prove physical truth. Proposed complementary checks include time and sequence validation, cross-sensor consistency, and operational plausibility limits derived from process physics. Their effectiveness depends on the process and on the independence of the available measurements.

4.4. Attack paths and common dependencies

Trace how one compromised component could affect the others. A shared credential, power supply, firmware image, or management network can defeat apparently independent safeguards. Review these dependencies before describing two monitoring components as independent. Include the verifier, update service, and recovery mechanism in the threat model rather than treating them as automatically trustworthy.

Time synchronization deserves particular attention because the freshness checks proposed in this paper depend on it. Input-age limits, evidence freshness, and certificate validity all assume a trustworthy clock. Attacks on the Network Time Protocol can shift a client's clock [33], and delay attacks against both NTP and the Precision Time Protocol can introduce offsets that authentication alone does not prevent [34]. A shared time source therefore appears in Fig. 6 as a common dependency of the gateway and the verifier. Mitigations include authenticated time distribution, monotonic sequence numbers and nonces that do not rely on wall-clock time, holdover monitoring that alarms on abrupt offsets, and cross-checking against an independent time reference.

5. A secure software foundation

The software layer defines who may access data, which artifacts may run, and how changes are approved. Joint guidance on secure AI development covers design, development, deployment, and operation [20]. The implementation below applies that lifecycle approach to the industrial monitoring service. These controls are necessary even when hardware protection is available, and at the advisory level they constitute most of the required protection (Fig. 2). They also define the objects and policies that hardware-backed mechanisms must enforce: the approved model and preprocessing pipeline, the allowable data sources, the execution version, and the permissions attached to each result. Hardware isolation without this software-defined context cannot establish that a useful or authorized computation was performed.

The lifecycle view in Fig. 4 makes the software–hardware dependencies visible before focusing on inference. At each handoff, the receiving stage must check the approved artifact and the accompanying evidence rather than merely trusting the previous stage's output. The figure is a proposed assurance workflow, not an implemented system.

5.1. Data and training controls

Maintain an inventory of data sources, permissions, transformations, and label owners. Separate training and evaluation data by appropriate time periods or sites to reduce leakage between datasets; random splits of correlated time series overstate performance. Preserve examples of rare but consequential operating conditions, such as start-up,

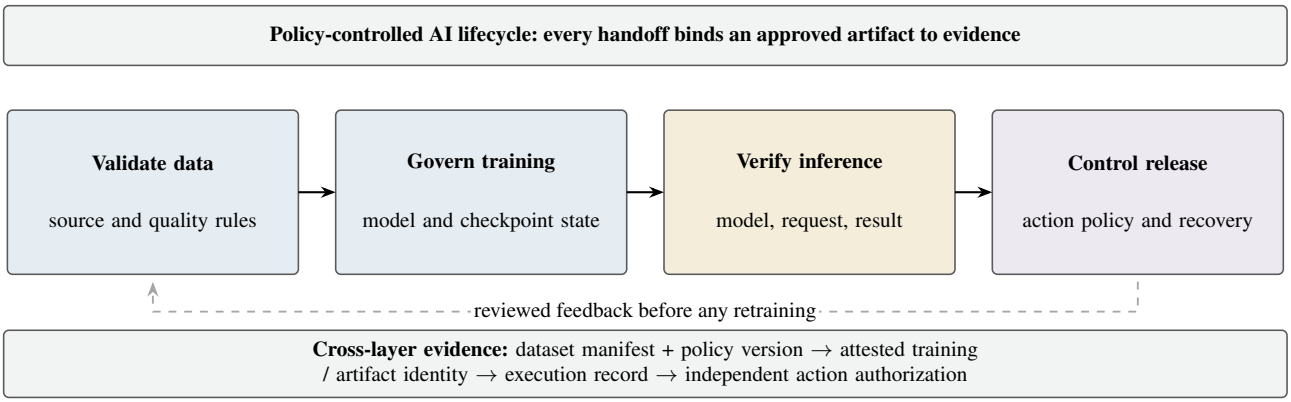


Figure 4. Proposed lifecycle assurance chain. Software defines admission and release policies; hardware-backed mechanisms can protect their execution, keys, and records. Valid evidence for a processing step does not prove that measurements or labels are physically correct. Feedback re-enters training only after review.

shutdown, and maintenance modes, which are often underrepresented. Record rejected data and unresolved quality issues rather than silently discarding them. Authorize changes to labels and feature extraction because both can change model behavior without changing the model architecture.

5.2. Approved-data validation and evidence binding

Before training, a proposed validation gateway would check whether each source and dataset snapshot is authorized for the stated use and whether the observations satisfy specified format, provenance, quality, temporal, and cross-sensor consistency rules. For industrial telemetry, these checks should account for operating modes, sensor calibration, and site or time coverage. Accepted records would be bound to a dataset manifest containing digests, source identifiers, validation-policy and preprocessing versions, review outcomes, and unresolved exceptions. Admission to the training pipeline would require approval of that particular manifest, not merely access to a general-purpose storage location.

The hardware–software co-design question is which parts of validation and manifest generation must execute in an isolated environment, which keys or measurements need hardware-backed protection, and what the training service must verify before releasing dataset-decryption keys. A TEE, TPM, or hardware security module may strengthen code and key isolation and attest the measured environment; hardware hashing or dedicated logic could accelerate selected integrity checks if justified by the workload. The verifier must still evaluate the attestation chain and bind its decision to the approved policy and dataset digest. A signed manifest or valid enclave measurement proves neither that a compromised sensor reported physical truth nor that labels are correct. Source authorization and process plausibility require external evidence and operational review. This is a proposed design pattern whose implementation cost and threat coverage remain to be evaluated (Section 8).

5.3. Build and supply-chain integrity

Protect the build and training environment with distinct service identities, restricted credentials, reviewed dependencies, and authenticated artifact storage. Bind each candidate to its dataset version, training configuration, preprocessing code, and evaluation report. Supply-chain frameworks provide the mechanisms: SLSA defines levels of build provenance and tamper resistance [35], and in-toto records signed attestations for each step of a pipeline so that a verifier can check that the steps were performed by the authorized parties in the authorized order [36]. Applied to AI, the provenance record should cover the dataset snapshot and training run as well as the code, because a substituted dataset produces a different model from identical code.

The approved artifact identifier and its preprocessing and numerical configuration must be available to the runtime verifier in Section 6.1; a signed build record alone cannot constrain a runtime that is free to load another artifact. Training approval should authorize a specific candidate, not every artifact produced by a trusted account. Secure checkpoints and export paths because they can expose model state or enable unauthorized substitution. For vendors, IEC 62443-4-1 [7] and the Cyber Resilience Act [26] make these practices part of the product's security obligations; for operators, NIS2 requires supply-chain risk to be managed [25].

5.4. Deployment and network separation

Place the analytics service in its own IEC 62443 zone, separate from both the control zone and the enterprise network, with conduits that carry only the required data flows [5]. Prefer passive or read-only collection for the first operational pilot, for example from a network tap or a data diode, so that the analytics zone has no path to send traffic into the control zone. Keep training infrastructure and general-purpose external services outside the control path. Authenticate administrative access and schedule changes with the operations team. The joint Deploying AI Systems Securely guidance addresses configuration and operation of

externally developed AI systems and calls for adaptation to the deployment's threat profile [21].

5.5. Inference and action authorization

Validate input format, time, sequence, source, and processing status before invoking the model. Use an approved model and feature pipeline. Record whether data were missing or outside the expected operating range. Route the result through a separate policy service that checks the requested action, affected asset, user authority, and current operating state. A model score alone should not be sufficient to authorize an operational change. This separation also defines the output-release interface for hardware protection: an execution result may be authenticated or verified, but the policy service independently decides whether that result is eligible for an action (Section 6.3).

5.6. Controlled feedback and model updates

Keep operational feedback in a review queue. Require approval before it enters retraining, and test the candidate against both routine and exceptional conditions. Unreviewed feedback is a poisoning channel: an attacker who can influence which alerts operators dismiss can gradually teach the model to ignore an attack pattern. Promote versions through a staging environment and an explicitly bounded pilot. Define how an operator can withdraw a model without also disabling essential monitoring or control. Automatic online learning remains outside the initial scope because it changes approved behavior during operation.

The baseline should also preserve useful audit records without unnecessarily copying sensitive telemetry. Log decisions, model versions, policy outcomes, and failures. Limit raw-data access and retention according to the operational purpose. The audit service must tolerate a failed AI service without losing the record of the failure, and should be write-once from the perspective of the analytics zone so that a compromised AI service cannot erase its own traces.

6. Hardware protection and verifiable execution

The software baseline in Section 5 specifies approved artifacts, data handling, and action permissions, but its enforcement can be undermined if the attacker controls the host that runs the AI service. Hardware support can reduce that exposure by isolating selected execution, protecting keys, and measuring the environment. A cross-layer design must bind these hardware measurements to software-defined identities and policies; attesting to a machine without establishing which model, feature pipeline, and output path it executes provides only a limited assurance claim. The design must still identify which firmware, runtime, drivers, and devices belong to its trusted computing base, meaning the components whose failure could invalidate the security claim.

Current CPU trusted execution environments isolate either an enclave within a process, as in Intel SGX [37], or an entire virtual machine from the hypervisor and host, as in AMD SEV-SNP [38] and Intel TDX [39]. Confidential virtual machines are the more practical target for AI services because existing frameworks and drivers run largely unmodified. Mo et al. systematize how these mechanisms have been applied to machine learning and the guarantees each provides [40]. Trusted execution does not remove side channels: transient-execution attacks such as Foreshadow have extracted secrets from SGX enclaves [41], and the platform's microcode and firmware update status is part of the security claim.

6.1. Attestation and protected key release

Remote attestation supplies evidence about an environment for assessment against a verifier's policy. RFC 9334 separates the attester, verifier, and relying party and addresses the freshness of evidence [27]. The proposed key service releases model or data keys only to an approved environment bound to the intended session. Check the evidence chain, approved versions, revocation status, and relevant configuration. A stale but validly signed record should not authorize a new session; freshness should rest on a verifier-supplied nonce rather than on the attester's clock (Section 4.4).

When inference uses both a CPU and an accelerator, the verifier needs composite evidence that covers both devices and the channel between them, and the policy must reject configurations in which either part is unverified. The software runtime must expose stable model, configuration, and request identifiers, while the platform must measure or otherwise enforce the corresponding code, device state, and transfer channel. Key release, artifact loading, and output eligibility are therefore linked design decisions rather than independent security features. Attestation cannot establish that the input reflects reality or that the measured software has no defects. Nor does a successful startup check rule out faults later in execution. The inference record must state the limited claim supported by the platform and runtime. A signature authenticates a statement; it becomes meaningful execution evidence only through an appropriate chain of measurement and enforcement.

Attestation is not continuous runtime assurance. A successful startup measurement can establish an initial state only within the platform's stated assumptions. A later fault in weights, intermediate activations, device memory, or output-release logic may still change the result. The proposed design therefore separates (i) pre-execution approval of the model and platform, (ii) runtime checks or protected fault-status signals where supported, and (iii) a per-request output-release decision that rejects absent, stale, or invalid evidence. These checks need not be implemented by the same device, but their evidence and trust dependencies must be made explicit. Section 8.3 outlines fault-coverage experiments; neither attestation nor a signature alone proves that a particular result was free of post-verification faults.

6.2. Accelerator design choices

One design includes the accelerator within the trusted boundary and protects transfers, memory, and checkpoint storage. IPU Trusted Extensions (ITX) is a published example of confidential computing extensions within an AI accelerator, supported by compiler and runtime changes [42]. Commercial GPUs now provide a similar capability: the NVIDIA H100 confidential computing mode extends the trust boundary to the GPU, with device attestation and encrypted transfers between the CPU's trusted environment and the GPU [43]. Another design delegates work to an untrusted accelerator and checks its results. Slalom demonstrates such a partition for neural network inference, outsourcing linear layers to an untrusted GPU and verifying them from within an enclave [44]. These are design foundations, not proof that a new platform meets the same guarantees.

These options create different software obligations. A protected accelerator may require changes to buffer management, driver interfaces, device initialization, and the evidence chain; verification of an untrusted accelerator instead requires a precisely specified reference computation and a verifier that can check delegated results. The partition of model operations is therefore chosen jointly with the trust boundary, memory movement, verification cost, and response-time budget (Section 8.1).

The architectural alternatives in Fig. 5 illustrate why the accelerator partition and the proof or attestation mechanism must be selected together. They establish different trust boundaries and therefore support different claims; neither route alone validates an operational decision.

Figure 5 distinguishes the state and transfers that each option must protect. In a protected-accelerator design, plaintext weights and activations reside only in memory covered by the approved CPU–device boundary. CPU–device transfers and direct memory access (DMA) permissions must restrict access to approved buffers. In a delegation design, the trusted verifier retains reference state, and the accelerator receives only the operands permitted by the selected protocol. The verifier checks returned values against the approved computation. Startup evidence covers the initial configuration. Detection of faults after startup requires separate runtime checks and an explicit fault model.

Choose between these routes based on the required confidentiality and integrity properties. An untrusted accelerator may see weights or intermediate values unless the selected protocol protects them. Result verification alone does not provide model confidentiality. Similarly, memory encryption does not automatically protect timing behavior, device interfaces, or the code responsible for releasing results. In edge OT deployments, the accelerator is often an FPGA or a small embedded processor rather than a data-center GPU, and the available protection mechanisms differ accordingly (Section 8.4).

6.3. Computation evidence and operational validity

Proof-based verification provides evidence that a specified computation was performed without trusting the executing platform. SafetyNets demonstrates this for a class of neural networks represented as arithmetic circuits [45]; later zero-knowledge systems such as [46], [47] and halo2-based constructions [48] extend proofs to convolutional networks at larger scale. Proof generation nonetheless remains substantially more expensive than native inference, which constrains its use in latency-bound control loops and makes it more suitable for after-the-fact audit or low-rate decisions.

Co-design also determines what is being verified. Quantization, accelerator arithmetic, and compiler transformations can change the relation between an approved mathematical model and its executed representation. A verification claim must name the numerical representation and allowable error, and the safety review must assess whether changes near an anomaly threshold alter missed-event risk (Section 8.2).

Such evidence addresses whether the specified computation was performed. A critical-system decision also needs evidence that the model, input, numerical representation, and resulting action are suitable for the operating context. These are separate assurance questions. Neither an attested runtime nor a computational proof establishes that isolating a controller is operationally safe.

The proposed deployment therefore verifies computation where required and then applies an independent action policy. If execution evidence is absent, invalid, or late, the AI output is ineligible for action. The physical system follows its approved degraded-operation procedure, which must be designed separately (Section 7.2).

7. Proposed architecture and safe degraded operation

The proposed architecture separates collection, AI processing, and action authorization. Its purpose is to prevent a single misleading prediction or compromised analytics service from directly determining a critical action. The initial use case is the industrial anomaly detector of Section 3.2 operating in advisory mode. Any later automation requires a separately approved change in authority and a revised assurance case. The architecture is also a co-design boundary map: software policy defines which model and action are approved; hardware-backed execution may provide evidence about how the model ran; and independent process protection constrains the physical consequences. No one of these layers substitutes for the others.

The binding across layers is explicit: a telemetry record identifies the input and its freshness, a model artifact identifies approved code and numerical settings, an execution record binds the result to the selected runtime, and an external policy evaluates the proposed action. The verifier must reject any missing binding rather than treating a successful signature or boot measurement as sufficient.

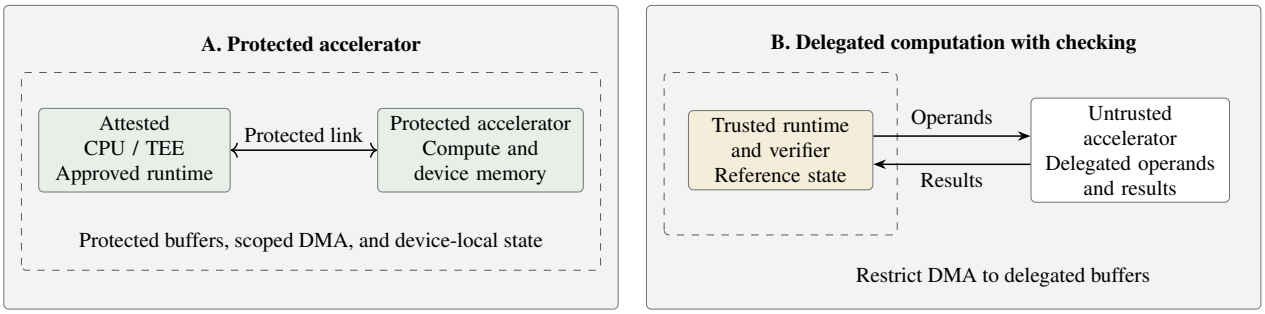


Figure 5. Two alternative CPU–accelerator trust boundaries, not interchangeable guarantees. (A) A protected accelerator extends the measured execution environment to a device and its transfers. (B) A trusted component verifies results from untrusted acceleration; result integrity does not by itself imply confidentiality. Both alternatives require application-specific performance and security evaluation.

TABLE 4. RESPONSIBILITIES AND TRUST BOUNDARIES IN THE PROPOSED ARCHITECTURE

Component	Proposed responsibility	Boundary to enforce
Telemetry gateway	Authenticate and validate incoming observations	No implied authority to change the process
Protected AI service	Run the approved model and feature pipeline	No direct actuator or administrative credentials
Evidence verifier	Check execution records and freshness	Reject invalid evidence independently of the model
Action policy service	Check operational limits and permissions	Only explicitly authorized actions can proceed
Independent protection	Maintain required process safety functions	AI failure must not disable required safeguards

Table 4 summarizes the responsibilities and boundaries of the proposed architecture, and Fig. 6 shows the components, zones, and trust boundaries.

7.1. A proposed transaction

(1) A telemetry gateway receives an observation and (2) records its source, timestamp, sequence, and quality flags. (3) The approved inference environment processes it and produces a result with a model and configuration identifier. Where required, the runtime attaches evidence that binds this request and output to the approved environment. (4) The verifier checks that record before the result reaches the action policy service. (5) An operator receives the supporting information and (6) authorizes any consequential intervention, which is carried out through conventional control within the approved workflow.

A candidate inference record binds the request identifier and input record to the approved preprocessing, model, and numerical configuration, together with the output and evidence for the selected execution boundary. The verifier checks that the evidence covers the required execution components, matches the current request, and arrives before the result expires. A failed check withholds the result from the action policy service and signals degraded operation. For an eligible result, the action policy service checks the proposed operation, affected asset, current process state, and assigned authority before it can be presented for operator authorization.

7.2. Failure handling depends on the process

If verification fails, withhold the AI-derived action and preserve the event for investigation. That does not mean every critical service should stop. Depending on the hazard analysis, the approved response might retain conventional control, switch to a limited operating mode, or follow a controlled shutdown sequence. The appropriate response cannot be generalized across a water plant, a railway system, and a hospital. A blanket network disconnect or immediate power-off may itself create harm. Fig. 7 summarizes the eligibility checks and the fallback path.

Specify behavior for lost telemetry, unavailable key services, delayed inference, failed updates, and cloud disconnection. Include a maximum permissible age for inputs and results. The operator should see that AI assistance is degraded instead of receiving an apparently normal output based on stale data. Recovery should require confirmation that the process state and the software state are consistent.

A fail-safe design creates a lever for the adversary. If invalid, missing, or late evidence makes the AI output ineligible, an attacker who can disrupt the verifier, the key service, the time source, or the network path can deliberately push the system into degraded mode and so switch off AI-based monitoring during an intrusion. The degraded state is safe for the process but not necessarily for the security posture. Three measures reduce this risk. Entry into degraded mode should itself raise an alert that reaches the security team, not only the operator. Repeated or correlated failures, particularly those coinciding with other anomalies, should be treated as a possible attack rather than as an availability

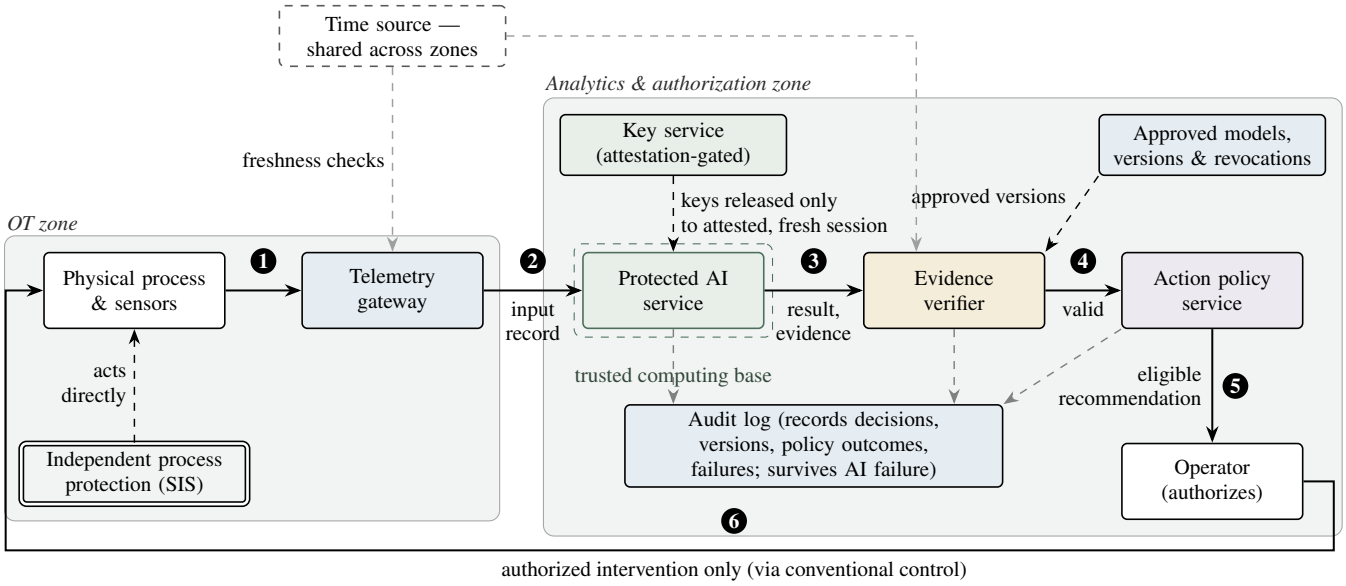


Figure 6. Proposed advisory monitoring architecture. Steps correspond to Section 7.1; AI components hold no actuator credentials, independent protection operates outside the AI path, and the shared time source remains a common dependency (Section 4.4).

issue. Finally, conventional monitoring that does not depend on the AI path should continue to operate in degraded mode, so that losing the AI reduces detection capability rather than removing it.

7.3. Independent oversight

Joint guidance on integrating AI into OT emphasizes governance, assurance, monitoring, and failsafe practices [22]. In this proposal, safeguards must have sufficient independence for the identified hazards, including relevant shared resources and administration. The safety instrumented system, designed under IEC 61511 [11], must not depend on the AI path for its inputs, its engineering access, or its configuration; TRITON shows that safety systems are themselves targets and that a shared engineering workstation can undermine their independence [3]. Merely deploying two copies of the same model does not establish independent protection, because both copies share the same training data, failure modes, and evasion vulnerabilities.

Human review is useful only if the operator has adequate information, time, training, and authority to intervene. Operators who work with a usually reliable automated aid tend to monitor it less closely and to accept its recommendations without sufficient verification, a pattern known as automation bias [49]. The AI Act requires high-risk systems to enable overseers to remain aware of this tendency [23]. Practical countermeasures include presenting the evidence behind each alert rather than a score alone, measuring how often operators override or confirm recommendations, and periodically injecting known test cases to check that review remains effective.

8. Hardware–software co-design priorities

Hardware–software co-design is not the sequential addition of hardware isolation to an already fixed software workflow. It jointly selects the software representation, hardware trust boundary, evidence mechanisms, and operational response so that the resulting system can support a stated assurance claim under its timing and recovery constraints. A design that secures only one layer may fail at another: encrypted memory does not validate sensor readings, a verified accelerator result does not authorize process control, and an approved model does not establish which device executed it. Figure 8 summarizes these dependencies visually, and Table 5 maps them to the work packages below. Together they form a design and evaluation framework, not a ranking of implemented platforms.

These interactions must be evaluated against an operational envelope defined in Section 3, not against throughput alone. The following work packages are proposed experiments, each with a research question and observable measures. Security, AI, and operational owners should agree on acceptance criteria before implementation (Section 9.3).

8.1. A protected inference and evidence path

Begin with one model, one processor–accelerator platform, and a documented operating envelope. Compare a software-only baseline, an attested CPU implementation, and an accelerator-backed design under the same inputs and threat assumptions; do not treat these variants as offering identical confidentiality or integrity claims. Co-design buffer placement, authenticated transfers, evidence generation, and

TABLE 5. CROSS-LAYER DESIGN DECISIONS AND EVALUATION QUESTIONS (PROPOSED TAXONOMY)

Design concern	Software-level decision	Hardware or execution decision	Joint assurance question and proposed measurement
Protected inference	Approved runtime, model identity, buffers, output release	CPU/accelerator trust boundary, encrypted transfers, attestation	Is the result bound to the approved execution path within the deadline? Measure end-to-end latency, rejected evidence, and resource contention.
Numerical representation	Quantization, thresholds, reference computation	Arithmetic format, kernels, proof or verification support	Does the executable representation preserve detection and verification guarantees? Measure event-level detection, approximation error, and verification cost.
Leakage and faults	Access patterns, recomputation, sensitive model operations	Isolation, memory protection, fault detection	Which observations and injected faults remain effective? Measure leakage exposure, warning suppression, and added latency.
Updates and recovery	Artifact provenance, version policy, rollback authorization	Secure loading, device keys, FPGA configuration, firmware recovery	Can a compromised or obsolete configuration be rejected while retaining authorized recovery? Test update interruption and downgrade attempts.
Data admission and training continuity	Dataset validation, manifest approval, candidate identity, checkpoints	Isolated validation/training, protected keys, attested loading	Can training be restricted to the approved dataset and the deployed model bound to the evaluated candidate? Test substitution, key release, and evidence continuity.

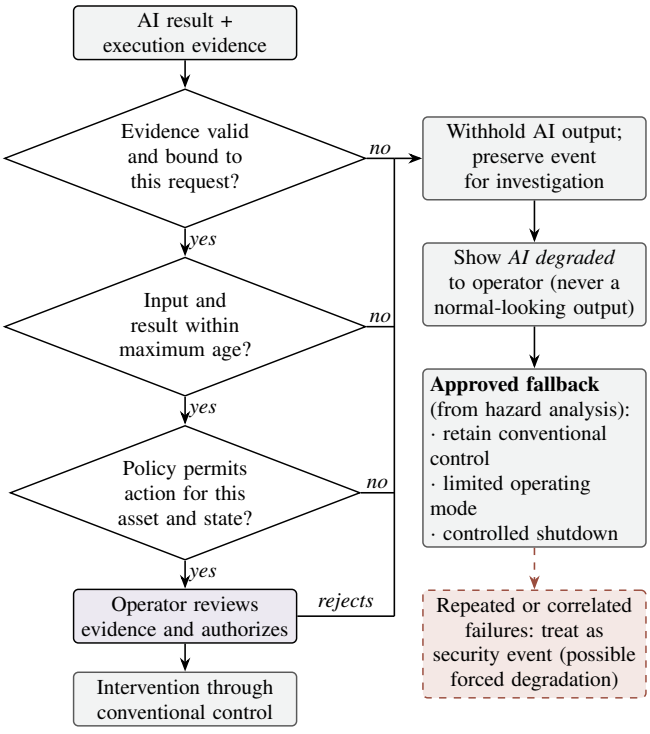


Figure 7. Output eligibility and degraded-operation logic. A failed check withholds the AI output; the approved fallback is hazard-dependent, and repeated or correlated failures may indicate an attack.

the output-release check. Measure the complete path from receipt of telemetry to delivery of an eligible result. Include queueing, key operations, encryption of CPU–accelerator transfers, and verification costs rather than reporting accelerator kernel time alone; for confidential GPU modes, transfer encryption rather than computation is a likely bottleneck [43]. Report latency as a distribution with tail percentiles and deadline misses, not as a mean. Keep control functions insulated from resource contention caused by analytics workloads.

Research question: for a given latency budget, which combination of protected execution and evidence generation provides the strongest integrity claim, and what is the cost of each component?

8.2. Numerical precision and verification

Investigate quantization or approximations only with a clearly specified reference computation. Here the co-design variable is the joint choice of model precision, hardware arithmetic, and the verification relation: a faster kernel or shorter proof is useful only if its numerical behavior remains acceptable for the detector and its operating envelope. Integer quantization is widely used to reduce inference cost [50], and many proof systems also require fixed-point or finite-field arithmetic [45], [46]. Reduced precision may change classification thresholds, particularly near the decision boundary, which is where an evasion attacker operates. Measure missed-event and false-alarm rates under the transformed model, and test whether quantization changes the model's vulnerability to adaptive evasion. If a proof or verification protocol operates over a different arithmetic representation, document the mapping and allowed error. Approve the numerical configuration together with the model instead of treating it as a transparent implementation detail.

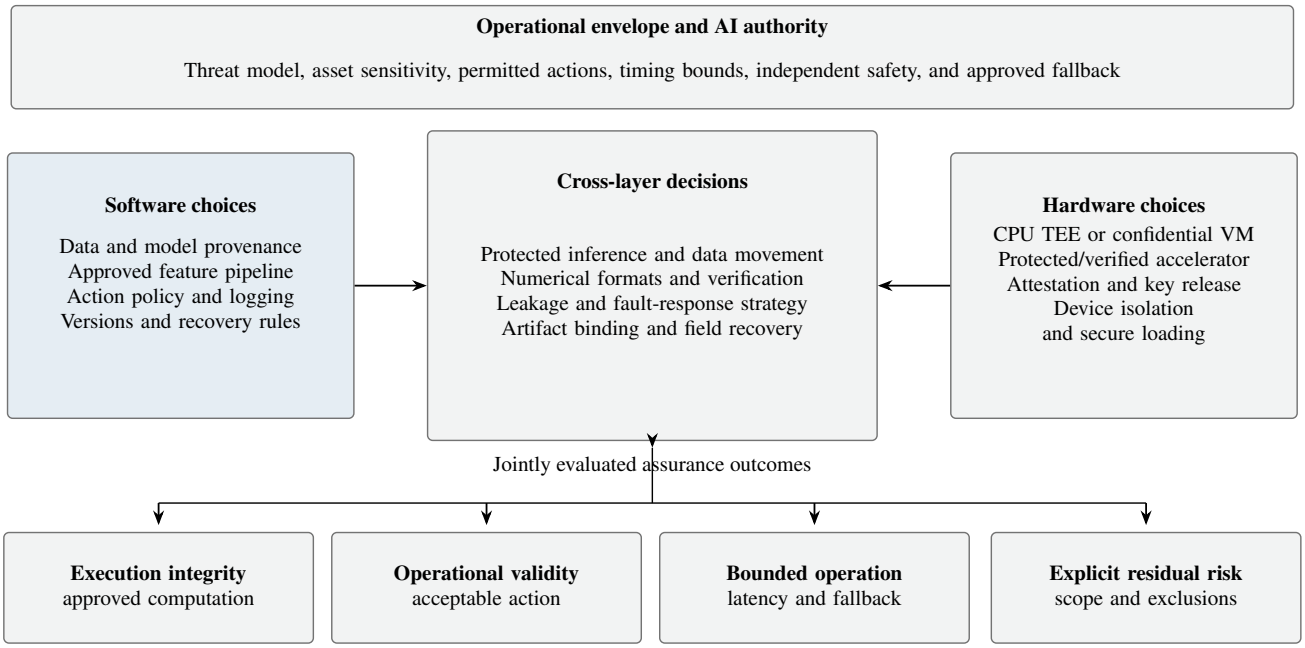


Figure 8. Cross-layer co-design taxonomy (proposed). AI authority and operating constraints shape the software and hardware choices; the coupled decisions must be evaluated for execution integrity, operational validity, bounded operation, and residual risk.

Research question: how much does each numerical transformation change detection quality and robustness, and can the verified computation be shown to match the approved reference within a stated error bound?

8.3. Leakage and fault resilience

Specify which observations the attacker can collect before evaluating side-channel defenses. The protection boundary includes software memory-access behavior and scheduling as well as hardware isolation and fault-detection features; evaluating either in isolation would miss cross-layer leakage or warning-suppression paths. Electromagnetic and timing side channels can reveal a neural network's architecture and parameters on embedded devices [51], and a recovered model enables offline evasion (Section 3.4). Candidate defenses include constant-time inference kernels, controlled memory access schedules, and masking of selected arithmetic.

Fault attacks are a distinct threat. A small number of targeted bit flips in stored weights can degrade a network to near-random accuracy [52], and many networks have individual parameters whose corruption causes large accuracy loss, which Rowhammer-style attacks can exploit [53]. For an anomaly detector, the harmful fault is one that suppresses a warning, not necessarily one that reduces average accuracy. Evaluate integrity checks on weights, error-correcting memory, recomputation, and redundant execution of sensitive operations. Distinguish random errors from targeted attacks. Protect the detector and output-release path, and test whether a fault can suppress a warning while allowing a corrupted result to pass.

Research question: which parameters and operations of the detector are most sensitive to targeted faults, and can selective protection of those elements provide most of the benefit of full redundancy at lower cost?

8.4. Device configuration and field recovery

A field-programmable gate array (FPGA) deployment requires protection of both model artifacts and the bitstream that configures the hardware. The software release manifest, device configuration, key-release policy, and permissible recovery version must refer to a consistent approved deployment state. Study authenticated and encrypted bitstream loading, protected device keys, controlled updates, and rejection of unauthorized older versions. NIST SP 800-193 describes firmware resilience through protection, detection, and recovery [28]. The Update Framework shows how update metadata can be designed to survive the compromise of individual signing keys and to prevent rollback and freeze attacks [54]. The proposed system should retain an authorized recovery route without permitting an attacker to select an obsolete vulnerable configuration. Recovery authority must be explicit and auditable. Vendors placing such products on the EU market must also provide security updates over the support period required by the Cyber Resilience Act [26], which favors update mechanisms designed for long equipment lifetimes.

8.5. Private training and evidence continuity

Protected inference begins with a model whose training history and state can be accounted for. For confidential

training, start with selected fine-tuning layers or critical kernels rather than assuming full-model training fits a protected processor–accelerator platform. Map the exposure of weights, gradients, activations, optimizer state, and checkpoints: shared gradients alone may reveal training examples [55]. Software choices include the trainable layers, precision, kernel schedule, data placement, checkpoint frequency, and export policy. Hardware choices include the execution boundary, protected buffers, memory and accelerator transfer protection, and evidence or key-release support. A plausible partition is one that satisfies the intended confidentiality claim without violating the available time, memory, or energy budgets; a TEE or accelerator does not provide all of these guarantees automatically.

Bind each training run to the validated dataset manifest of Section 5.2, approved model initialization, code and configuration versions, and evaluation result. The distinction between auditable training and mere hardware isolation also arises in cryptographically verified training: VeriDP proves specified differentially private SGD operations in zero knowledge, illustrating how a software-defined training claim can be checked without equating it with data correctness [56]. This is a different assurance mechanism from attestation and remains a research alternative rather than a required component of the proposed OT deployment. The key cross-layer handoff is from an authorized software training record to enforced artifact loading: neither a signed manifest nor an attested device by itself demonstrates that the evaluated candidate is the one actually executing. An attested runtime must enforce the model identity and reject unauthorized substitutes at deployment. The SENTRA system [57] illustrates protected training with trusted execution, secret sharing, collective attestation, and versioned state under node churn; it does not establish suitability for latency-bound OT inference without separate validation.

Research question: which partition of training kernels, protected state, accelerator transfers, and checkpoint operations provides the intended confidentiality and traceability guarantees at acceptable training time, energy, memory, and communication cost? Compare against an explicitly defined software baseline, and report model quality and residual leakage alongside system costs.

The expected research output is a comparison of candidate designs reporting measured trade-offs and explicit residual risks, rather than a single optimized performance figure. The project should report when a hardware feature improves isolation but fails a latency budget, when a numerical change harms detection, or when a safeguard cannot cover the selected fault model.

8.6. Distributed inference for multi-antenna radio frequency sensing

Figure 9 shows the distributed radio frequency (RF) sensing architecture, the second case used to evaluate the co-design framework. Each antenna node retains its raw samples and uses a local multiprocessor and general matrix multiplication (GEMM) engine. A multiprocessor (MP) hub collects protected contributions and a central neural processing unit (NPU) fuses them into one result. For the same sensing task and model, compare two compute partitions. In partition A, the antenna nodes extract features and the central NPU executes the model. In partition B, the antenna nodes also execute selected model layers, and the central NPU executes the remaining layers. In both partitions, the central NPU fuses the contributions. The partition determines where weights and intermediate representations are exposed, what crosses the links, and which compute engines, shared memory, and direct memory access (DMA) paths require isolation or verification.

The design should protect sensor data and model weights across the selected partition. Compare protected execution with delegated computation and result checking where the latter can also protect sensitive operands under the stated threat model. The root of trust should support measurement of node firmware, accelerator configuration, and approved model and preprocessing artifacts. Per-request evidence should bind the fused output to the sensing window, the identities and contributions of participating nodes, and the approved model and numerical configuration. Define the permitted contribution age, synchronization error, and required node set. Authentication establishes the source of a contribution. Signal quality and suitability for the sensing task require separate checks.

When a required contribution is missing, stale, or outside the synchronization limit, withhold the result and signal degraded operation. Any permitted reduced-node mode should be evaluated separately for detection, false alarms, and timing. Begin with advisory output and route any consequential action through independent authorization.

Layer partitioning across edge and central compute has been studied for latency and energy trade-offs [58]. Figure 10 compares how partitions A and B allocate processing within the architecture in Figure 9. The protected-execution and delegated-checking alternatives follow the ITX and Slalom precedents discussed in Section 6.2. RFC 9334, Section 10, provides approaches for checking the freshness of attestation evidence. The RF verifier must also enforce the application-specific rules for sensing windows and node participation.

Figure 11 details a candidate protected NPU for the central fusion stage in Figure 9. The host submits a job descriptor through a command interface that checks the approved workload before dispatch. The contribution ingress authenticates each node contribution and checks its sensing-window identifier against the approved job. Contributions that fail either check are excluded before DMA to the fusion buffers, and the ingress reports the rejection. The DMA interface restricts accepted transfers to buffers assigned to the job, and the GEMM engines access weights and activations in local memory. The result and status interface binds the output to the job and accepted contributions and supplies available fault and rejection status for external verification. The verifier checks that the required node contributions are present before authorizing advisory release. These fields contribute to the execution evidence described

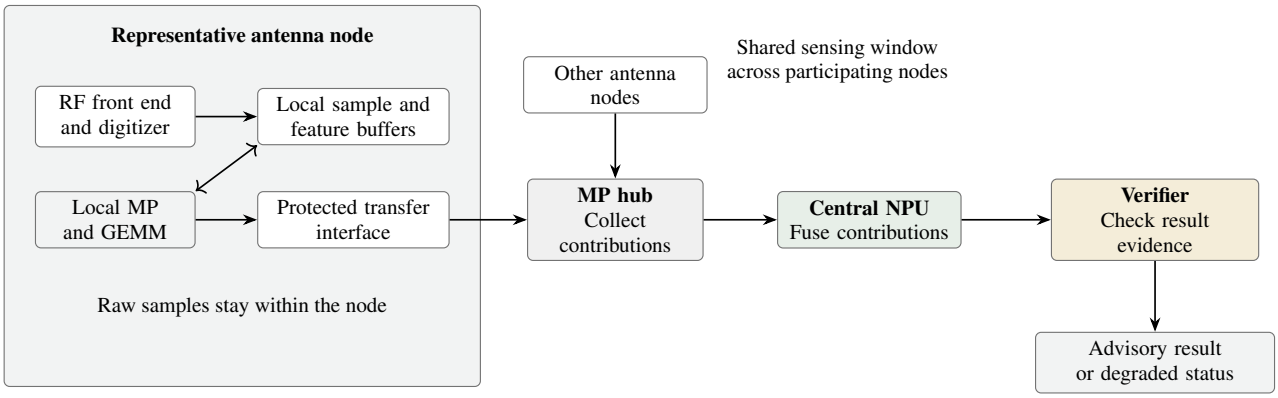


Figure 9. Proposed distributed RF sensing architecture. The expanded antenna node shows local acquisition, buffering, processing, and the protected transfer interface. The verifier checks that the fused result is bound to the sensing window, required node contributions, and approved configuration before releasing an advisory result or reporting degraded status.

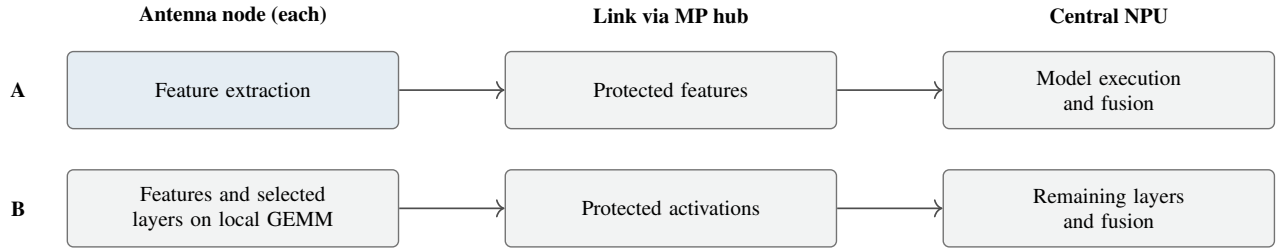


Figure 10. Proposed compute partitions for multi-antenna RF inference. Partition A sends protected features to the central NPU. Partition B executes selected model layers at the antenna nodes and sends protected activations.

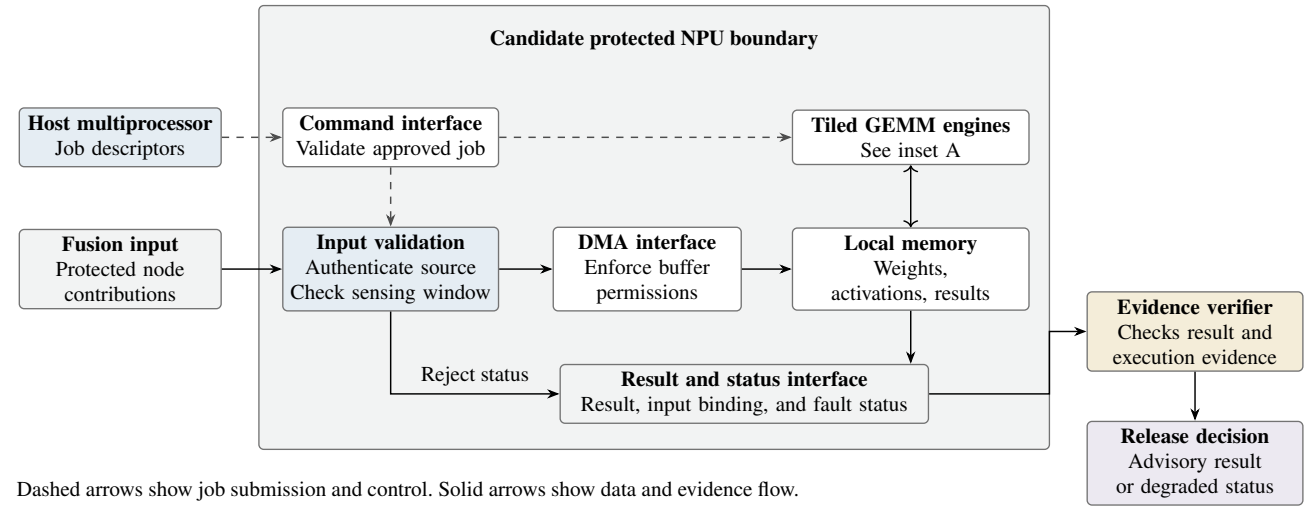
in Section 7.1. The chiplet inset illustrates one possible implementation. A chiplet is a die designed to operate with other dies in the same package. The chiplet controller implements RISC-V, an open instruction set architecture that specifies the instructions its processor executes [59]. It dispatches work over an on-chip network to tiles, which are repeated compute blocks. Each tile includes a scratchpad, a local memory whose contents are explicitly managed, and a GEMM array that performs matrix multiplication. A direct memory access (DMA) engine moves data between memory buffers without the host processor copying it. Gemmini provides a published example of an accelerator with a matrix array, scratchpad, and DMA engine [60]. One way to restrict device access to memory is an input/output memory management unit (IOMMU), which translates device addresses and enforces access permissions [61]. An inter-chiplet fabric is a link network connecting multiple dies in the package. UCIe is one published specification for such links [62]. These references describe implementation precedents for individual components of the candidate design.

Prior work on trustworthy platforms integrates hardware isolation, runtime, and operating-system design under latency constraints [63]. A candidate device architecture for physical AI combines compute cores, AI and cryptographic accelerators, trust anchors, and sensor and actuator interfaces under an operating system that limits the actions available to the AI. The proposed RF demonstrator exercises the sensing and inference path and releases advisory results. A later actuation prototype should treat commands proposed by local AI or a cloud service as requests subject to device-side authorization. Where cloud services participate, the device should authenticate the remote endpoint, protect the confidentiality and integrity of the channel, and check command freshness before applying its local authorization policy. The operating system should define the permitted operations and conditions, and the actuator interface should enforce the authorization decision. Accelerator access to memory and device interfaces must be constrained so that a command cannot bypass this check. The required protection functions should remain available when the AI service or cloud connection is unavailable.

Research question: Which antenna-to-NPU partition meets the confidentiality, execution-integrity, and detection requirements within the end-to-end response-time budget?

9. Validation and adoption considerations

The proposed architecture requires validation before operational use. Assessment should establish whether its protections address the identified threats, preserve useful detection performance, and meet the process's timing and recovery requirements. Where hardware protection is introduced, compare it with the secure software baseline under equivalent workloads and operating conditions. Report



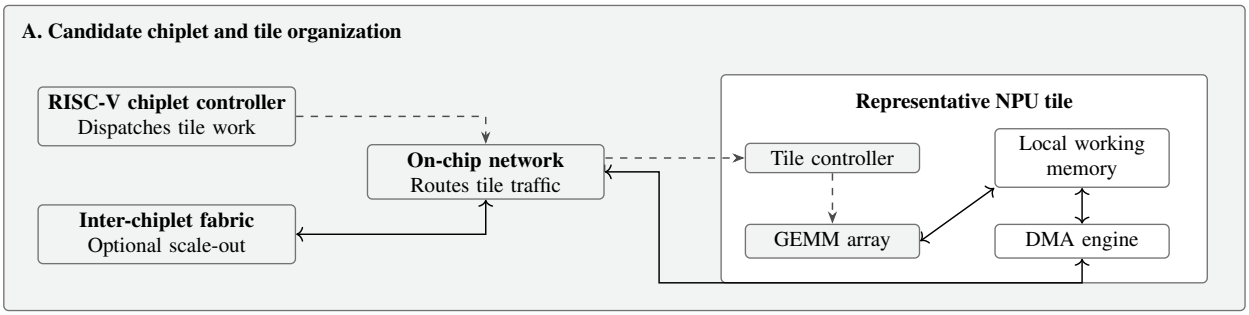


Figure 11. Candidate central NPU datapath for distributed RF inference. Job submission and dispatch are shown separately from the data path. The ingress authenticates contributions and checks their sensing window before DMA into job buffers. An external verifier checks the result evidence before advisory release.

changes in model behavior or numerical precision separately from protection overheads. Where possible, use an ablation matrix that separately enables software controls, protected execution, accelerator verification, and integrated output-policy enforcement; this distinguishes the benefit of each mechanism from the interactions among them. Such comparisons remain proposed evaluation procedures, not measured results of this white paper.

9.1. Evidence required before deployment

Validation should cover cybersecurity, detection quality, end-to-end latency, fault handling, and operational assurance. Relevant evidence includes resistance to artifact substitution and stale evidence; missed critical events and false alarms per operating period; deadline misses; recovery under partial service loss; and enforcement of action limits and approved fallback behavior. Tests should also establish whether an attacker can deliberately force degraded operation and suppress monitoring.

For the new dataset-admission and training path, test unauthorized or replaced dataset snapshots, invalid policy versions, altered labels, replayed manifests, key-release failures, unapproved checkpoint export, and model substitution at deployment. Report admission errors, false rejection of legitimate data, evidence-verification cost, training overhead, and the fraction of cases that require manual review. For runtime integrity, inject faults after successful startup attestation and test whether the output-release path rejects a corrupted or unverifiable result; measure undetected warning suppression, not just average classification accuracy. Experiments should distinguish guarantees supported by hardware measurements from properties that still require external process evidence.

Public datasets and testbeds can support initial assessment, but they do not establish suitability for a particular deployment. Evaluation should include representative operating conditions, exceptional process states, and adaptive attacks against the selected detector [30]. Detection metrics should reflect operational consequences: report event-level detection, detection delay, and false alarms per shift or day, rather than relying on point-adjusted scores that can overstate performance [64].

Figure 12 links the comparison of compute partitions to the decision to advance to shadow mode.

For the distributed RF case in Section 8.6, compare both partitions with the secure software baseline on the same inputs and operating conditions. Record latency from

the close of the sensing window to verified output release, including communication, queueing, and verification. Report tail latency, deadline misses, link traffic, and energy alongside event-level detection and false alarms. Evaluate changes in precision separately, particularly their effect near the detection threshold. For each partition, check whether an observer of the links or a component outside the approved trust boundary can obtain sensitive features, intermediate activations, or model weights under the stated threat model.

Replay a valid earlier contribution, introduce clock skew, remove a required node, substitute a model or device configuration, and inject faults after successful attestation. Measure invalid outputs released, valid outputs withheld, undetected suppression of detections, and time to report degraded operation. Interrupt a node update and attempt an unauthorized downgrade. Verify that the node can rejoin fusion only after its recovered configuration has been approved.

DeepSense 6G provides candidate real-world multimodal workloads [65]. Confirm that a selected scenario has synchronized contributions and event labels suitable for the fusion experiment, and use a controlled RF testbed for fault and attack injection.

For a later prototype with actuation authority, evaluate command authorization separately from inference integrity. Submit an out-of-scope command from an approved runtime, replay an authenticated command that has expired, and attempt to access the actuator interface through an accelerator or DMA path. Observe the actuator interface to determine whether any unauthorized command is executed. Measure the latency of authorized commands under representative contention, and verify that loss of connectivity or authorization produces the defined fallback while required protection functions remain available. Treat these tests as a separate gate before granting actuation authority.

Tie the acceptance tests to the interfaces in Figs. 5 and 9. For the protected-accelerator path, attempt an out-of-range DMA transaction and check that the target buffer is neither disclosed nor modified. For delegated computation, substitute an incorrect result and measure whether the verifier releases it, reporting the detection coverage for the selected checking method. For distributed RF inference, replay an authenticated contribution from a required node in an earlier sensing window while withholding its current contribution. Check that the ingress in Fig. 11 rejects the replay before it can write to the fusion buffer, that the verifier withholds the advisory result for the configured node set, and that degraded status is reported within the agreed deadline.

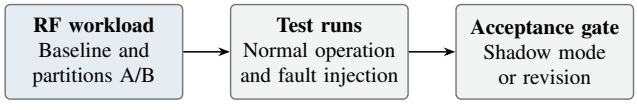


Figure 12. Proposed validation path for distributed RF inference. Fault-injected runs include replay, clock skew, node loss, and configuration faults. Advancement requires meeting the predefined criteria for detection, timing, confidentiality, and failure handling.

9.2. Staged adoption

Fig. 13 summarizes a staged adoption path. Begin by identifying assets, data flows, threats, operational hazards, and the functions that must continue without AI. Validate the secure software baseline in a laboratory, simulation, or approved test environment, then assess the selected hardware protections and their failure paths.

A shadow-mode pilot should precede operational reliance: the model observes the process without controlling it, allowing recommendations to be compared with operator decisions. Bounded advisory use may follow when the agreed criteria are met. Granting automated action authority requires a separate assessment and approval.

9.3. Go/no-go criteria

Before testing, operational, security, AI, and safety or assurance owners should agree on acceptance criteria and responsibilities. These criteria should cover response time, detection performance, attacker capabilities, independent protection, operator oversight, and fallback and recovery behavior. Thresholds must follow from the application and its hazard analysis.

For the industrial monitor in Section 3.2, the operations, security, and safety leads review the shadow-mode results against agreed limits for missed critical events, false alarms per shift, and late recommendations. They also check that the monitor withholds recommendations based on stale telemetry and that operators can continue monitoring when AI assistance is unavailable. If any mandatory criterion fails, the monitor stays in shadow mode.

Progression should depend on evidence that all mandatory criteria are met. If a criterion fails, retain the previously approved authority level or revise the design. High average detection accuracy alone is insufficient justification for deployment or increased authority.

9.4. Research roadmap for the RF–NPU demonstrator

The proposed research contribution is a protected inference demonstrator that connects distributed RF sensing to a central neural processing unit (NPU) and an external verifier. The sensing, accelerator, and verification groups should agree on the interfaces and acceptance criteria before selecting a hardware partition. Table 6 summarizes the milestones and the evidence required to advance.

10. Limitations and open questions

The architecture and roadmap have not been implemented or measured, and the protection levels in Fig. 2 are a proposal to be tested against real hazard analyses rather than a normative mapping. Several questions remain open. The proposed cross-layer taxonomy is an organizing framework, not evidence that the suggested combinations are feasible on

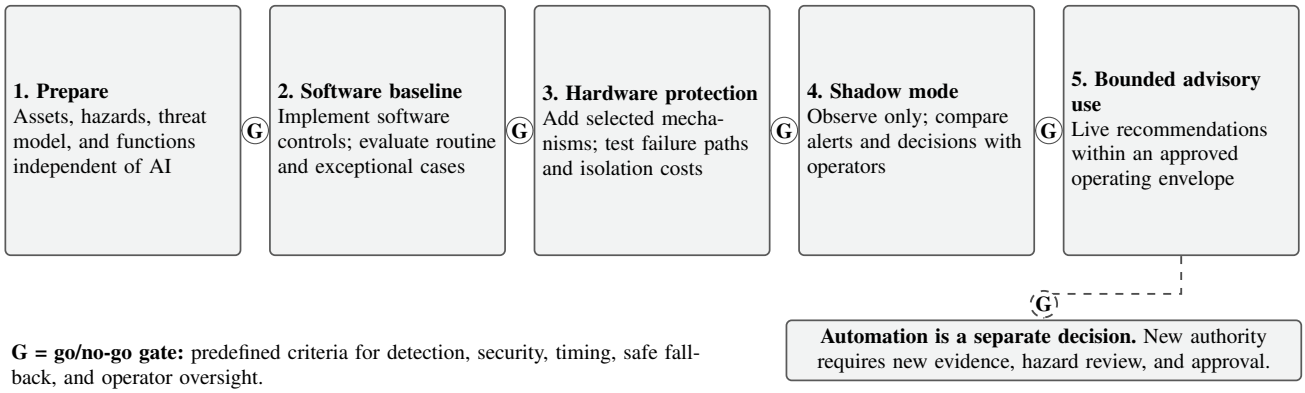


Figure 13. Staged adoption with go/no-go gates. Failed gates require revision or retention of the preceding approved authority level; automation needs separate authorization.

TABLE 6. PROPOSED RESEARCH ROADMAP FOR THE RF–NPU DEMONSTRATOR.

Milestone	Team deliverable	Evidence to proceed
1. Define interfaces	One RF sensing task, a software baseline, and agreed interfaces for sensing, acceleration, and verification.	Agree on required node inputs, evidence fields, detection and confidentiality criteria, deadlines, and degraded behavior.
2. Compare and prototype	Compare partitions A and B on the same workload. Prototype the selected partition.	Meet detection and latency limits. Block stale inputs before they reach fusion buffers and confine DMA to approved memory.
3. Validate the demonstrator	An integrated advisory prototype connected to the external verifier.	Release eligible results on time. For missing, stale, or invalid required inputs, withhold the result and report degraded status on time. Meet confidentiality and recovery criteria before shadow mode.

a particular OT platform. No measured end-to-end latency, fault coverage, or security gain is claimed.

First, there is no accepted method for expressing the failure behavior of a machine learning component in the terms that functional safety standards require, which is why this paper keeps AI outside safety functions. Second, the cost of verifiable inference remains high relative to the latency budgets of many control loops; it is unclear for which decisions after-the-fact verification is an acceptable substitute. Third, trusted execution for embedded and FPGA-based accelerators used at the OT edge is less mature than for data-center CPUs and GPUs. Fourth, the interaction between regulation and authority levels will become clearer only as guidance on the AI Act's high-risk classification and harmonized standards are published. Finally, the human-oversight measures proposed here need empirical evaluation with operators under realistic workload.

11. Conclusion

Secure AI for critical systems requires a protected lifecycle and a controlled path from prediction to action. Software defines policy and governs change. Hardware can strengthen execution protection and support verifiable records. Co-design is the explicit selection of trust boundaries, model representations, data transfers, verification paths, and recovery mechanisms under a common threat model and operational budget. Operational safeguards determine whether the wider system remains within acceptable limits when those components fail, including when an attacker causes them to fail on purpose. The protection required should follow from the authority the AI is granted, while confidentiality follows from the value of the assets and the evasion risk. The practical next step is a narrow, advisory prototype that demonstrates traceability, measured protection, and tested degraded operation before any expansion of autonomy.

Acknowledgments

Maryam Zarezadeh received funding from the German Federal Ministry of Research, Technology, and Space under the SEMECO-Q1 project (Grant No. 03ZU2210AA). Additional support came from funds allocated by Germany's Saxon State Parliament.

References

[1] R. Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011.

[2] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the Ukrainian power grid: Defense use case," Electricity Information Sharing and Analysis Center (E-ISAC) and SANS Industrial Control Systems, Tech. Rep., Mar. 2016.

[3] A. Di Pinto, Y. Dragoni, and A. Carcano, "TRITON: The first ICS cyber attack on safety instrument systems," Black Hat USA, 2018.

[4] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J.-F. Crespo, and D. Dennison, "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems*, vol. 28, 2015. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html

[5] International Electrotechnical Commission, "IEC 62443-3-2:2020 security for industrial automation and control systems – part 3-2: Security risk assessment for system design," International standard, 2020.

[6] ——, "IEC 62443-3-3:2013 industrial communication networks – network and system security – part 3-3: System security requirements and security levels," International standard, 2013.

[7] ——, "IEC 62443-4-1:2018 security for industrial automation and control systems – part 4-1: Secure product development lifecycle requirements," International standard, 2018.

[8] K. Stouffer, M. Pease, C. Tang, T. Zimmerman, V. Pillitteri, S. Lightman, A. Hahn, S. Saravia, A. Sherule, and M. Thompson, "Guide to operational technology (OT) security," National Institute of Standards and Technology, NIST Special Publication 800-82 Rev. 3, Sep. 2023. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-82r3

[9] K. Stouffer, M. Pease, C. Tang, A. Hahn, J. Gilsinn, D. Rebori-Carretero, O. Alexander *et al.*, "Guide to operational technology (OT) security," National Institute of Standards and Technology, NIST Special Publication 800-82 Rev. 4 (Initial Public Draft), Sep. 2026, comments due 30 Nov. 2026. [Online]. Available: https://csrc.nist.gov/pubs/sp/800/82/r4/ipd

[10] International Electrotechnical Commission, "IEC 61508:2010 functional safety of electrical/electronic/programmable electronic safety-related systems," International standard, Parts 1–7, 2010.

[11] ——, "IEC 61511:2016 functional safety – safety instrumented systems for the process industry sector," International standard, Parts 1–3, 2016, including Amendment 1:2017 to Part 1.

[12] ——, "IEC TR 63069:2019 industrial-process measurement, control and automation – framework for functional safety and security," Technical report, 2019.

[13] N. G. Leveson, *Engineering a Safer World: Systems Thinking Applied to Safety*. Cambridge, MA: MIT Press, 2011.

[14] E. Tabassi, "Artificial intelligence risk management framework (AI RMF 1.0)," National Institute of Standards and Technology, NIST AI 100-1, Jan. 2023. [Online]. Available: https://doi.org/10.6028/NIST.AI.100-1

[15] K. Megas, B. Cuthill, M. Dotter, M. Garris, I. Khemani, B. Patrick, N. Schiro, J. N. Snyder, and M. Zarei, "Cybersecurity framework profile for artificial intelligence (Cyber AI Profile): NIST community profile," National Institute of Standards and Technology, NIST Interagency Report 8596, Dec. 2025, initial preliminary draft. [Online]. Available: https://csrc.nist.gov/pubs/ir/8596/iprd

[16] International Organization for Standardization and International Electrotechnical Commission, "ISO/IEC 42001:2023 information technology – artificial intelligence – management system," International standard, 2023.

[17] A. Vassilev, A. Oprea, A. Fordyce, H. Anderson, X. Davies, and M. Hamin, "Adversarial machine learning: A taxonomy and terminology of attacks and mitigations," National Institute of Standards and Technology, NIST AI 100-2 E2025, Mar. 2025. [Online]. Available: https://doi.org/10.6028/NIST.AI.100-2e2025

[18] MITRE Corporation, "ATLAS: Adversarial threat landscape for artificial-intelligence systems," Knowledge base, https://atlas.mitre.org, 2026, accessed Oct. 2026.

[19] OWASP Foundation, "OWASP top 10 for large language model applications 2025," OWASP GenAI Security Project, https://genai.owasp.org, Nov. 2024.

[20] National Cyber Security Centre, Cybersecurity and Infrastructure Security Agency, and International Partners, "Guidelines for secure AI system development," National Cyber Security Centre, Tech. Rep., Nov. 2023. [Online]. Available: https://www.ncsc.gov.uk/collection/guidelines-secure-ai-system-development

[21] National Security Agency, Cybersecurity and Infrastructure Security Agency, Federal Bureau of Investigation, and International Partners, "Deploying AI systems securely," National Security Agency, Cybersecurity Information Sheet, Apr. 2024. [Online]. Available: https://media.defense.gov/2024/Apr/15/2003439257/-1/-1/0/CSI-DEPLOYING-AI-SYSTEMS-SECURELY.PDF

[22] Cybersecurity and Infrastructure Security Agency, Australian Signals Directorate's Australian Cyber Security Centre, National Security Agency's Artificial Intelligence Security Center, Federal Bureau of Investigation, Canadian Centre for Cyber Security, German Federal Office for Information Security, Netherlands National Cyber Security Centre, New Zealand National Cyber Security Centre, and United Kingdom National Cyber Security Centre, "Principles for the secure integration of artificial intelligence in operational technology," Cybersecurity and Infrastructure Security Agency and International Partners, Tech. Rep., Dec. 2025. [Online]. Available: https://www.fbi.gov/file-repository/cyber-alerts/principles-for-the-secure-integration-of-artificial-intelligence-in-operational-technology.pdf

[23] European Parliament and Council of the European Union, "Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence (artificial intelligence act)," Official Journal of the European Union, L 2024/1689, Jul. 2024. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2024/1689/oj

[24] ——, "Regulation (EU) 2026/1744 amending regulations (EU) 2024/1689, (EU) 2018/1139 and (EU) 2023/1230 as regards the simplification of the implementation of harmonised rules on artificial intelligence (digital omnibus on AI)," Official Journal of the European Union, L 2026/1744, Jul. 2026. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=OJ:L_202601744

[25] ——, "Directive (EU) 2022/2555 on measures for a high common level of cybersecurity across the union (NIS 2 directive)," Official Journal of the European Union, L 333, Dec. 2022. [Online]. Available: https://eur-lex.europa.eu/eli/dir/2022/2555/oj

[26] ——, "Regulation (EU) 2024/2847 on horizontal cybersecurity requirements for products with digital elements (cyber resilience act)," Official Journal of the European Union, L 2024/2847, Nov. 2024. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2024/2847/oj

[27] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, "Remote ATtestation procedureS (RATS) architecture," Internet Engineering Task Force, RFC 9334, Jan. 2023, informational. [Online]. Available: https://www.rfc-editor.org/rfc/rfc9334.html

[28] A. Regenscheid, "Platform firmware resiliency guidelines," National Institute of Standards and Technology, NIST Special Publication 800-193, May 2018. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-193

[29] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *25th USENIX Security Symposium*, 2016, pp. 601–618.

[30] A. Erba, R. Taormina, S. Galelli, M. Pogliani, M. Carminati, S. Zanero, and N. O. Tippenhauer, "Constrained concealment attacks against reconstruction-based anomaly detectors in industrial control systems," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2020, pp. 480–495.

[31] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.

[32] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection," in *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2023, pp. 79–90.

[33] A. Malhotra, I. E. Cohen, E. Brakke, and S. Goldberg, "Attacking the network time protocol," in *Network and Distributed System Security Symposium (NDSS)*, 2016.

[34] M. Ullmann and M. Vögeler, "Delay attacks – implication on NTP and PTP time synchronization," in *International Symposium on Precision Clock Synchronization for Measurement, Control and Communication (ISPCS)*, 2009, pp. 1–6.

[35] Open Source Security Foundation, "Supply-chain levels for software

artifacts (SLSA) specification v1.0," https://slsa.dev, Apr. 2023.
[36] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Cappos, "in-toto: Providing farm-to-table guarantees for bits and bytes," in *28th USENIX Security Symposium*, 2019, pp. 1393–1410.
[37] V. Costan and S. Devadas, "Intel SGX explained," Cryptology ePrint Archive, Paper 2016/086, 2016.
[38] Advanced Micro Devices, "AMD SEV-SNP: Strengthening VM isolation with integrity protection and more," White paper, Jan. 2020.
[39] P.-C. Cheng, W. Ozga, E. Valdez, S. Ahmed, Z. Gu, H. Jamjoom, H. Franke, and J. Bottomley, "Intel TDX demystified: A top-down approach," *ACM Computing Surveys*, vol. 56, no. 9, 2024.
[40] F. Mo, Z. Tarkhani, and H. Haddadi, "Machine learning with confidential computing: A systematization of knowledge," *ACM Computing Surveys*, vol. 56, no. 11, 2024.
[41] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution," in *27th USENIX Security Symposium*, 2018, pp. 991–1008.
[42] K. Vaswani, S. Volos, C. Fournet, A. N. Diaz, K. Gordon, B. Vembu, S. Webster, D. Chisnall, S. Kulkarni, G. Cunningham, R. Osborne, and D. Wilkinson, "Confidential computing within an AI accelerator," in *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. USENIX Association, Jul. 2023, pp. 501–518. [Online]. Available: https://www.usenix.org/conference/atc23/presentation/vaswani
[43] G. Dhanuskodi, S. Guha, V. Krishnan, A. Manjunatha, M. O'Connor, R. Nertney, and P. Rogers, "Creating the first confidential GPUs," *Communications of the ACM*, vol. 67, no. 1, 2024, also in ACM Queue 21(4), 2023.
[44] F. Tramèr and D. Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://arxiv.org/abs/1806.03287
[45] Z. Ghodsi, T. Gu, and S. Garg, "SafetyNets: Verifiable execution of deep neural networks on an untrusted cloud," in *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Online]. Available: https://arxiv.org/abs/1706.10268
[46] T. Liu, X. Xie, and Y. Zhang, "zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2021, pp. 2968–2985.
[47] F. Zarinjouei, M. Zarezadeh, B. Abdolmaleki, B. Mohee, A. Abidin, and S. Köpsell, "zkrnn: Zero-knowledge proofs for recurrent neural network inference," *Cryptology ePrint Archive*, 2026.
[48] D. Kang, T. Hashimoto, I. Stoica, and Y. Sun, "Scaling up trustless DNN inference with zero-knowledge proofs," arXiv:2210.08674, 2022.
[49] R. Parasuraman and D. H. Manzey, "Complacency and bias in human use of automation: An attentional integration," *Human Factors*, vol. 52, no. 3, pp. 381–410, 2010.
[50] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713.
[51] L. Batina, S. Bhasin, D. Jap, and S. Picek, "CSI NN: Reverse engineering of neural network architectures through electromagnetic side channel," in *28th USENIX Security Symposium*, 2019, pp. 515–532.
[52] A. S. Rakin, Z. He, and D. Fan, "Bit-flip attack: Crushing neural network with progressive bit search," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 1211–1220.
[53] S. Hong, P. Frigo, Y. Kaya, C. Giuffrida, and T. Dumitraş, "Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks," in *28th USENIX Security Symposium*, 2019, pp. 497–514.
[54] J. Samuel, N. Mathewson, J. Cappos, and R. Dingledine, "Survivable key compromise in software update systems," in *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, 2010, pp. 61–72.
[55] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.
[56] B. Abdolmaleki, A. R. Asadi, V. R. Asadi, S. Köpsell, B. Mohee, N. Roustaeifar, and M. Zarezadeh, "VeriDP: Verifiable differentially private training," *Proceedings on Privacy Enhancing Technologies*, vol. 2026, no. 3, pp. 48–64, 2026.
[57] M. Zarezadeh, J. Eisoldt, B. Mohee, S. Köpsell, and B. Abdolmaleki, "SENTRA: Privacy-preserving training in outsourced cloud environments," Cryptology ePrint Archive, Paper 2026/1443, 2026. [Online]. Available: https://eprint.iacr.org/2026/1443
[58] Y. Kang, J. Hauswald, C. Gao, A. Rovinski, T. Mudge, J. Mars, and L. Tang, "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge," in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2017, pp. 615–629.
[59] *The RISC-V Instruction Set Manual, Volume I: Unprivileged Architecture*, RISC-V International, 2024. [Online]. Available: https://docs.riscv.org/reference/isa/v20240411/unpriv/intro.html
[60] H. Genc, S. Kim, A. Amid *et al.*, "Gemmini: Enabling systematic deep-learning architecture evaluation via full-stack integration," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 2021.
[61] *RISC-V IOMMU Architecture Specification*, RISC-V International, 2025. [Online]. Available: https://docs.riscv.org/reference/iommu/v20250620/iommu_intro.html
[62] *Universal Chiplet Interconnect Express (UCIe) Specification*, UCIe Consortium. [Online]. Available: https://www.uciexpress.org/specifications
[63] F. Pauls, S. Haas, S. Köpsell, M. Roitzsch, N. Asmussen, and G. Fettweis, "On trustworthy scalable hardware/software platform design," in *2022 Smart Systems Integration (SSI)*, 2022.
[64] S. Kim, K. Choi, H.-S. Choi, B. Lee, and S. Yoon, "Towards a rigorous evaluation of time-series anomaly detection," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 7194–7201.
[65] A. Alkhateeb, G. Charan, T. Osman, A. Hredzak, J. Morais, U. Demirhan, and N. Srinivas, "DeepSense 6G: A large-scale real-world multi-modal sensing and communication dataset," *IEEE Communications Magazine*, vol. 61, no. 9, pp. 122–128, 2023.