

SBOM $\leftrightarrow$ Runtime Concordance Vector (SRCV): Semantic Proof of Build Integrity for NASA JWST Science Platforms

Bee Rosa Davis

NASA Johnson Space Center

Flight Operations Directorate (FOD)

2101 NASA Parkway, Houston, TX 77058

bee.r.davis@nasa.gov

bee_davis@alumni.brown.edu

Abstract—Scientific data pipelines that process observations from the James Webb Space Telescope (JWST) depend on strict reproducibility and auditable provenance across heterogeneous stacks (bare-metal HPC nodes, containerized services, and firmware-controlled platforms). Conventional SBOM checks rely on brittle string matching and cannot distinguish legitimate drift (e.g., approved patch updates) from high-risk divergence (e.g., unsigned kernel modules, shadowed system libraries, or misreported BMC/UEFI firmware). We introduce the SBOM $\leftrightarrow$ Runtime Concordance Vector (SRCV), a semantic verification framework that aligns a signed “golden image” SBOM with observed runtime artifacts via learned multi-channel embeddings and a two-stage matcher.

SRCV encodes component identity using structured features (vendor/product subwords, semantic version tuples, signature-chain metadata, cryptographic hashes, path context, and lineage) and incorporates runtime-only signals (loader precedence, overlay activity, TPM/IMA attestation, firmware trust tier). Similarities between declared and observed sets are computed as M_{ij} , summarized into coverage C , with policy-relevant penalties for extraneous artifacts E , unsigned/unknowns U , and anchor-quality gaps Q . A linear, interpretable decision score, $\text{SRCV}_{\text{raw}} = aC - bE - cU - dQ - \sum_{\ell \in \text{critical}} \eta_{\ell} \mathbb{1}[\text{sim}_{\ell} < \tau_{\ell}]$, drives an admission/provisioning gate implemented as policy-as-code and produces a one-click, auditor-ready evidence diff.

We present an architecture tailored to NASA science platforms: a shared-bottom, dual-head metric model; sparse K-NN + Sinkhorn soft assignment with bounded complexity and numeric safeguards; priority-ordered hard matching for gating; firmware trust tiers; runtime-only artifact allowlists; and cross-enclave, privacy-preserving outbreak detection via similarity sketches. We outline calibration (ROC/PR cost optimization with isotonic probability mapping), performance budgets (provisioning $p95 \leq 3$ s), and evaluation with red-team injects and cross-enclave generalization. SRCV operationalizes “IaC with embedded security” for JWST by making runtime concordance *measurable, explainable, and enforceable* from BIOS/UEFI up through container layers.

Index Terms—SBOM, supply chain security, runtime attestation, embeddings, JWST, NASA, OPA/Rego, Redfish, TPM, IMA/EVM, firmware trust tiers.

I. INTRODUCTION

Scientific data pipelines that transform raw observations from the James Webb Space Telescope (JWST) into calibrated, science-ready products operate under strict require-

ments for reproducibility, provenance, and change control. These pipelines span heterogeneous compute substrates—bare-metal high performance computing (HPC) nodes, containerized microservices, and firmware-governed control planes—and they must deliver deterministic results across proposal cycles and reprocessing campaigns. Ensuring that *what is running* faithfully reflects the signed, approved “golden” build is therefore a first-order concern for both science integrity and Authority-to-Operate (ATO) compliance.

Conventional verification relies on Software Bill of Materials (SBOM) string matching at build or deploy time. While necessary, string-level equality checks are brittle in practice: they struggle to distinguish legitimate, pre-approved drift (e.g., a patch-level CUDA driver update) from high-risk divergence (e.g., an unsigned kernel module, a shadowed `libc`, or Baseboard Management Controller (BMC) firmware reporting an incorrect family). They also provide little graded assurance: a single mismatch can force an all-or-nothing decision without explainability or class-aware risk treatment.

SBOM $\leftrightarrow$ Runtime Concordance Vector (SRCV) addresses these gaps by replacing brittle equality checks with a *semantic* comparison between the signed SBOM of the golden image and the *observed* runtime state. SRCV represents both declared and observed components via multi-channel embeddings (vendor/product subwords, semantic version tuples, signature-chain features, cryptographic hash sketches, filesystem path context, and lineage), then performs a two-stage alignment: a sparse K-NN + Sinkhorn soft assignment for stable coverage estimates and neighbor explanations, followed by a priority-ordered hard matching step that produces crisp policy semantics (missing, extraneous, unsigned) for gating.

Formally, let $S = \{s_i\}$ be SBOM-declared components and $R = \{r_j\}$ be observed runtime artifacts (OS packages, loaded kernel/GPU modules, container layers/overlays, and firmware identifiers). Let f_S, f_R be learned encoders and define

$$M_{ij} = \cos(f_S(s_i), f_R(r_j)). \quad (1)$$

SRCV summarizes alignment quality via (i) a class-weighted coverage term $C = \frac{\sum_i w_i \max_j M_{ij}}{\sum_i w_i}$; and (ii) penalties for

extraneous artifacts E (unmatched r_j), unsigned/unknown components U , and anchor-quality gaps Q (e.g., missing TP-M/IMA attestation or low firmware trust tier). An interpretable linear decision score then drives policy:

$$\text{SRCV}_{\text{raw}} = aC - bE - cU - dQ - \sum_{\ell \in \mathcal{C}_{\text{crit}}} \eta_{\ell} \mathbf{1}[\text{sim}_{\ell} < \tau_{\ell}], \quad (2)$$

with non-negative coefficients ($a, b, c, d, \eta_{\ell} \geq 0$) and class-specific similarity floors τ_{ℓ} for critical components (kernel, GPU drivers, firmware/BMC, `libc`). A monotone isotonic mapping is used for probability displays only; thresholds are set on SRCV_{raw} for interpretability.

A. Motivation: JWST Platform Realities

JWST workloads mix (i) bare-metal nodes with strict kernel/driver controls, (ii) containerized services where overlay semantics and loader precedence can shadow system libraries, and (iii) firmware surfaces (UEFI/BMC) with inconsistent telemetry quality across vendors. Reproducibility demands that pipelines run on known-good stacks during proposal deadlines and reprocessing windows; security and compliance demand auditor-ready evidence explaining any deviation.

B. Limitations of String-Match SBOM Checks

String equality fails gracefully neither for benign drift nor for subtle attack surfaces. Examples include:

- **Legitimate change:** a minor *signed* GPU driver update in an approved maintenance window; string mismatch should not hard-fail the node.
- **High-risk drift:** an *unsigned* kernel module or a container overlay shadowing `libc`; these require immediate block with explainable reasons.
- **Firmware ambiguity:** BMC/UEFI version strings without attestation; requires graded trust rather than binary pass/fail.

C. SRCV at a Glance

SRCV introduces:

- 1) **Shared-bottom, dual-head metric model** that encodes declared and observed components, augmenting the runtime head with loader/overlay/attestation signals.
- 2) **Two-stage matching**—sparse K-NN + Sinkhorn for stable coverage/explanations, then priority-ordered hard matching for policy semantics.
- 3) **Interpretable score** (Eq. 2) with class-aware thresholds learned under cost constraints (ROC/PR operating points) and guardrails on critical false negatives.
- 4) **Operational guardrails** including firmware trust tiers, runtime-only artifact (ROA) allowlists, policy-as-code gating, one-click evidence diffs, and fallbacks/degraded modes with explicit flags.

D. Contributions

This paper contributes:

- A *semantic* verification framework that grades concordance between SBOM and runtime, rather than relying on brittle equality.
- A scalable alignment procedure with numeric safeguards and time-bounded fallbacks compatible with provisioning SLOs ($p95 \leq 3$ s).
- A NASA-tailored operationalization: firmware trust tiers; ROA governance; auditor-ready evidence diffs; and a policy-as-code gate for both bare-metal provisioning and cluster admission.
- An evaluation plan for JWST enclaves including mimicry injects, cross-enclave generalization, attention stability checks, and quarterly red-team exercises.

E. Threat Model and Assumptions

We assume an adversary capable of introducing modules, overlays, or images with misleading vendor/version strings and partial signature chains. We *do not* assume the ability to forge cryptographic anchors (hashes, trusted signer roots) or TPM quotes. Firmware telemetry may be incomplete; enforcement follows a trust-tier schedule that begins in observe-only mode.

F. Paper Roadmap

Section II formalizes SRCV notation and the scoring model. Section III details the embedding architecture and runtime-state features. Section IV presents the alignment algorithm and complexity bounds. Section V describes the policy gate and evidence diff. Section VI covers calibration, SLOs, and evaluation. Section VII discusses adversarial robustness, firmware tiers, and limitations; Section VIII concludes.

II. SRCV FORMALISM AND DECISION RULE

A. Artifacts, Classes, and Sets

Let $S = \{s_i\}_{i=1}^{|S|}$ denote *declared* components from the signed golden-image SBOM, and $R = \{r_j\}_{j=1}^{|R|}$ the *observed* runtime artifacts (OS packages, loaded kernel/GPU modules, container layers/overlays, and firmware identifiers). Each artifact $x \in S \cup R$ is associated with a class $c(x) \in \mathcal{C}$, where

$$\mathcal{C} = \{\text{kernel}, \text{gpu_driver}, \text{firmware}, \text{bmc}, \text{libc}, \text{base_layer}, \text{library}, \text{tool}\}. \quad (3)$$

Let $\mathcal{C}_{\text{crit}} \subset \mathcal{C}$ denote the *critical* subset (e.g., `kernel`, `gpu_driver`, `firmware`, `bmc`, `libc`). We write $\mathcal{A}$ for the signed *runtime-only artifact (ROA) allowlist* associated with the workload class (e.g., JIT caches, ephemeral attest blobs). Artifacts in $\mathcal{A}$ are excluded from extraneous penalties (but still logged and evaluated for anomalous structure).

B. Encoders and Similarity

Let $f_S : S \rightarrow \mathbb{R}^d$ and $f_R : R \rightarrow \mathbb{R}^d$ be learned encoders that map declared and observed artifacts into a common metric space (details in Sec. III). We L2-normalize outputs so that cosine similarity

$$M_{ij} = \cos(f_S(s_i), f_R(r_j)) \in [-1, 1] \quad (4)$$

is well-defined. For coverage calculations we use a nonnegative similarity $\widetilde{M}_{ij} = (1 + M_{ij})/2 \in [0, 1]$ to avoid negative contributions.

Alignment placeholders: SRCV employs a two-stage alignment (sparse K-NN + Sinkhorn soft assign, then priority-ordered hard matching). We introduce here the quantities required by the score; the concrete alignment procedure and complexity bounds are given in Sec. IV.

C. Coverage and Penalties

Each declared component s_i has a nonnegative class weight $w_i \equiv w(c(s_i))$ (critical > important > peripheral) with $\sum_i w_i > 0$. The *class-weighted coverage* of the declared set by the observed set is

$$C = \frac{\sum_{i=1}^{|S|} w_i \max_{1 \leq j \leq |R|} \widetilde{M}_{ij}}{\sum_{i=1}^{|S|} w_i} \in [0, 1]. \quad (5)$$

Let π denote the stage-2 hard assignment mapping declared components to observed artifacts (or $\emptyset$ if unmatched), and define the unmatched observed set $\mathcal{U}_R = \{r_j \in R : \nexists i \text{ s.t. } \pi(s_i) = r_j\}$. Let $\kappa(\cdot) \geq 0$ be a class severity for extraneous artifacts and $v(\cdot) \geq 0$ a class severity for unsigned/unknown artifacts. We define:

$$E = \frac{1}{|R|} \sum_{r_j \in \mathcal{U}_R \setminus \mathcal{A}} \kappa(c(r_j)), \quad (6)$$

$$U = \frac{1}{Z_U} \sum_{r_j \in R} \mathbf{1}[\text{unsigned/unknown}(r_j)] v(c(r_j)), \quad (7)$$

where Z_U normalizes U to $[0, 1]$ (e.g., $Z_U = \sum_{r_j \in R} v(c(r_j))$). Anchor/firmware quality is captured by

$$Q = \phi_{\text{att}} q_{\text{att}} + \phi_{\text{fw}} q_{\text{fw}}, \quad (8)$$

with $q_{\text{att}} \in [0, 1]$ penalizing missing/weak TPM/IMA evidence and $q_{\text{fw}} \in [0, 1]$ penalizing low firmware trust tiers (T2 > T1 > T0). The coefficients $\phi_{\text{att}}, \phi_{\text{fw}} \geq 0$ are calibrated per enclave.

D. Decision Score and Class Floors

Let $\tau_\ell \in [0, 1]$ be a *similarity floor* for each critical class $\ell \in \mathcal{C}_{\text{crit}}$, evaluated against the matched pair produced by stage-2 hard assignment. The SRCV decision score is

$$\text{SRCV}_{\text{raw}} = aC - bE - cU - dQ - \sum_{\ell \in \mathcal{C}_{\text{crit}}} \eta_\ell \mathbf{1}[\text{sim}_\ell < \tau_\ell], \text{family}. \quad (9)$$

with nonnegative coefficients $a, b, c, d, \eta_\ell \geq 0$. Here $\text{sim}_\ell \in [0, 1]$ denotes the observed similarity (from $\widetilde{M}$) for the matched critical component of class ℓ ; if the component is missing or unsigned, the indicator fires by definition.

Monotonicity: Given fixed $(w_i, \kappa, v, \phi, \eta_\ell, \tau_\ell)$, SRCV_{raw} is monotone increasing in C and monotone decreasing in E, U, Q , and it strictly decreases when any critical class violates its floor.

E. Gating and Probability Display

Operational gating is applied to SRCV_{raw} and the class floors:

- **Hard fail:** if any $\text{sim}_\ell < \tau_\ell$ or any critical artifact is unsigned/unknown.
- **Score threshold:** else compare SRCV_{raw} against an enclave-specific cutoff t^* learned from ROC/PR cost optimization with critical false-negative guardrails.

For dashboards, a monotone isotonic map $\hat{P}(\text{allow} | \text{SRCV}_{\text{raw}})$ provides calibrated probabilities and precomputed 95% bootstrap confidence intervals per score bucket; thresholds remain on SRCV_{raw} for interpretability.

F. Notation Summary

TABLE I
KEY SYMBOLS USED IN SRCV FORMALISM.

Symbol	Meaning
S, R	Declared SBOM set; observed runtime set
$\mathcal{C}, \mathcal{C}_{\text{crit}}$	All classes; critical class subset
f_S, f_R	Encoders for declared/observed artifacts ($\mathbb{R}^d$)
$M_{ij}, \widetilde{M}_{ij}$	Cosine similarity; nonnegative similarity $(1+M)/2$
w_i	Class weight for s_i (critical > important > peripheral)
C	Class-weighted coverage (Eq. 5)
E, U, Q	Extraneous, unsigned/unknown, and anchor/firmware penalties
$\kappa(\cdot), v(\cdot)$	Class severities for E and U
τ_ℓ, η_ℓ	Similarity floor and penalty for critical class ℓ
SRCV_{raw}	Decision score (Eq. 9)
t^*	Learned enclave-wide threshold on SRCV_{raw}

Remark (ROA expiry). ROA allowlists are signed, versioned, and time-boxed. If expired at evaluation, the artifact is treated as quarantined with a 24 h grace for renewal; critical workloads may request auto-renewal via ticketed justification. The event is recorded in the evidence diff.

III. EMBEDDING ARCHITECTURE AND RUNTIME-STATE FEATURES

A. Feature Channels and Tokenization

Each artifact $x \in S \cup R$ exposes structured fields:

$$x = (v(x), p(x), \nu(x), c(x), \sigma(x), h(x), \xi(x), \lambda(x)),$$

for vendor v , product p , semantic version ν , class c , signature-chain metadata σ , cryptographic hash h , filesystem path context ξ , and lineage λ (e.g., container base ancestry or firmware

$e_{\text{txt}} = E_{\text{txt}}([v, p])$	(subword / hashed n -gram encoding)
$e_{\text{ver}} = E_{\text{ver}}(\nu)$	(semver tuple: major, minor, patch, prerelease flags)
$e_{\text{cat}} = E_{\text{cat}}(c)$	(learned categorical embedding)
$e_{\text{sig}} = E_{\text{sig}}(\sigma)$	(validity bit, chain depth, issuer CN/SKI sketch)
$e_{\text{hash}} = E_{\text{hash}}(h)$	(count-sketch/Bloom-style digest sketch)
$e_{\text{path}} = E_{\text{path}}(\xi)$	(char n -grams + privileged-dir indicators)
$e_{\text{lin}} = E_{\text{lin}}(\lambda)$	(graph features: depth, parent-digest match, family)

Each channel is LayerNormed and subject to channel dropout (probability $p \approx 0.1$) to prevent dominance by any single feature family.

B. Shared-Bottom, Dual-Head Metric Model

A shared encoder $\psi(\cdot; \theta_0)$ ingests the common channels $\{e_{\text{txt}}, e_{\text{ver}}, e_{\text{cat}}, e_{\text{sig}}, e_{\text{hash}}, e_{\text{path}}, e_{\text{lin}}\}$ and produces a fused representation via learned cross-channel attention (Sec. III-C). The declared/runtime heads then specialize:

$$f_S(s) = \text{MLP}_S(\psi(s; \theta_0); \theta_S), \quad (10)$$

$$f_R(r) = \text{MLP}_R([\psi(r; \theta_0), E_{\text{state}}(r)]; \theta_R), \quad (11)$$

with an ℓ_2 penalty on $\|\theta_S - \theta_R\|_2$ to discourage gratuitous divergence. Outputs are ℓ_2 -normalized so that the cosine similarity in Eq. (4) is well-defined. Unless stated otherwise, we use $d=256$ and 2-layer MLP heads with hidden widths $\{512, 256\}$ and GELU activations.

C. Cross-Channel Attention and Stability

Let $\mathcal{K}$ be the set of available channels for the artifact ($\mathcal{K}_S = \{\text{txt}, \text{ver}, \text{cat}, \text{sig}, \text{hash}, \text{path}, \text{lin}\}$ and $\mathcal{K}_R = \mathcal{K}_S \cup \{\text{state}\}$). For each $k \in \mathcal{K}$,

$$\alpha_k = \frac{\exp(u^\top \tanh(W_k e_k + b_k))}{\sum_{m \in \mathcal{K}} \exp(u^\top \tanh(W_m e_m + b_m))}, \quad (12)$$

$$z = \sum_{k \in \mathcal{K}} \alpha_k e_k, \quad \psi(\cdot) = \text{MLP}_0(z; \theta_0). \quad (13)$$

Initialization and training safeguards. Attention is initialized uniform; during the first epoch we apply temperature annealing ($T:5 \rightarrow 1$) and an entropy penalty $\lambda_{\text{ent}} H(\alpha)$ to avoid early collapse. **Inference-time recovery.** If $H(\alpha)$ falls below a floor, we set $\alpha_k \leftarrow 1/|\mathcal{K}|$ (uniform), flag the sample, and emit the reason code `attention_collapse` in the evidence diff.

D. Runtime-State Dictionary and Encoding

$E_{\text{state}}(r)$ adds runtime-only signals (absent from SBOM) as normalized numeric/ordinal/one-hot features. We list the core fields and encodings; all features are LayerNormed.

TABLE II
RUNTIME-STATE FEATURES $E_{\text{state}}(r)$ AND ENCODINGS.

Feature	Type	Encoding
ld_preload_present	bool	$\{0, 1\}$
ld_library_path_delta	int	clipped count (numeric)
link_precedence_hit	bool	$\{0, 1\}$
rpath_write_since_boot	bool	$\{0, 1\}$
taint_flags	bitmask	8-dim one-hot
module_load_time_s	numeric	log-scaled
module_signed	bool	$\{0, 1\}$
signer_tier	ordinal	$\{0:\text{none}, 1:\text{self}, 2:\text{int}, 3:\text{root}\}$
secure_boot_active	bool	$\{0, 1\}$
overlayfs_layers	int	numeric
overlay_write_set_kb	numeric	log-scaled
overlay_touch_paths	multi-hot	dirs: /usr/lib, /lib64, ...
container_privileged	bool	$\{0, 1\}$
rootless	bool	$\{0, 1\}$
ima_policy	ordinal	$\{0:\text{disabled}, 1:\text{audit}, 2:\text{appraise}, 3:\text{enforce}\}$
tpm_quote_ok	bool	$\{0, 1\}$
firmware_trust_tier	ordinal	$\{2:T2, 1:T1, 0:T0\}$
redfish_consistency_score	numeric	$[0, 1]$
overlay_risk	numeric	XGBoost score $[0, 1]$

ROA policy. Runtime-only artifacts approved by the signed, time-boxed ROA allowlist are excluded from E (extraneous) but still logged; expired allowlists trigger quarantine with a 24 h grace (Sec. II).

E. Training Objective and Hard Negatives

We train f_S, f_R with a contrastive objective and auxiliary heads for regularization.

Contrastive loss: Given minibatch pairs $\mathcal{B} = \{(s_i, r_i^+)\}$ from golden deployments/approved windows and in-batch negatives $\mathcal{N}_i = \{r_j : j \neq i\}$, we use InfoNCE:

$$\mathcal{L}_{\text{NCE}} = - \sum_{i \in \mathcal{B}} \log \frac{\exp(\cos(f_S(s_i), f_R(r_i^+))/\tau)}{\sum_{r \in \{r_i^+ \} \cup \mathcal{N}_i} \exp(\cos(f_S(s_i), f_R(r))/\tau)}. \quad (14)$$

We apply symmetric NCE (swap roles of S and R) for stability.

Auxiliary tasks: A multi-task head predicts (i) class c and (ii) signature validity (valid/invalid/unknown), yielding $\mathcal{L}_{\text{aux}}$; we also regularize attention via $\lambda_{\text{ent}} H(\alpha)$ in early training.

Gradient-aware hard negatives (text) and structural negatives: We generate adversarial text/version negatives by Projected Gradient Descent (PGD) on token embeddings with L_∞ radius $\epsilon=10^{-2}$ (validated by a human plausibility study), constrained by vendor/semver regexes. Structural hard negatives preserve anchor conflicts: signer-nullification, lineage family swaps, and digest-family collisions. The overall loss:

$$\mathcal{L} = \mathcal{L}_{\text{NCE}} + \lambda_{\text{aux}} \mathcal{L}_{\text{aux}} + \lambda_{\text{ent}} H(\alpha). \quad (15)$$

Algorithm 1 SRCV Training (contrastive + adversarial negatives)

- 1: Sample minibatch $\mathcal{B}$ of (s_i, r_i^+) from approved windows
 - 2: Build in-batch negatives $\mathcal{N}_i$ and generate PGD text/version negatives respecting regex
 - 3: Add structural negatives (signer-null, lineage swaps) with anchor conflicts
 - 4: Compute f_S, f_R via shared-bottom dual-head with attention; cache $\psi(\cdot)$
 - 5: Compute $\mathcal{L}_{\text{NCE}}, \mathcal{L}_{\text{aux}}$, and entropy term; update $\theta_0, \theta_S, \theta_R$
 - 6: Log attention weights α , monitor $H(\alpha)$ and gradient norms for collapse
-

F. Inference and Caching

We precompute and cache $f_S(s)$ for golden images and frequently observed components; $f_R(r)$ is computed on demand with $E_{\text{state}}(r)$. Attention weights α and per-channel contributions are logged and surfaced in the evidence diff for explainability. This section interfaces with alignment in Sec. IV, which details sparse K-NN construction, Sinkhorn bounds, and the hard-matching protocol.

IV. ALIGNMENT ALGORITHM AND COMPLEXITY BUDGET

SRCV aligns declared components S to observed runtime artifacts R in two stages: (i) a *soft* assignment over a sparse candidate graph to produce stable coverage/explanations; and (ii) a *hard* priority-ordered matching that feeds policy semantics (missing/extraneous/unsigned) and the score in Eq. (9). This section details the algorithmic steps, numeric guardrails, and complexity envelopes that meet the provisioning SLO.

A. Stage 0: Precomputation and Indexing

We precompute and cache encodings for declared components, $f_S(s_i)$, and compute $f_R(r_j)$ on demand (Sec. III). For each evaluation:

- 1) Build or refresh an ANN index (e.g., HNSW) over $\{f_R(r_j)\}$ on the target node/enclave.
- 2) Optionally, maintain a small cache of frequent f_R for recurring artifacts (e.g., common base layers).

B. Stage 1: Candidate Retrieval (Sparse K-NN)

For each $s_i \in S$, query the ANN index to obtain a candidate set $\mathcal{N}_k(i) \subseteq \{1, \dots, |R|\}$ with $|\mathcal{N}_k(i)| = k$. Define a sparse similarity matrix on this bipartite graph:

$$\widetilde{M}_{ij} = \frac{1 + \cos(f_S(s_i), f_R(r_j))}{2} \quad \text{for } j \in \mathcal{N}_k(i), \quad (16)$$

and $\widetilde{M}_{ij} = 0$ otherwise. Typical $k \in [16, 32]$ balances recall and runtime.

C. Stage 1b: Soft Assignment via Sparse Sinkhorn

We compute a *partial* transport plan $P \in \mathbb{R}_{\geq 0}^{|S| \times |R|}$ supported on the sparse graph that approximates the best alignment while allowing unmatched mass (rows/columns need not sum to 1). Let $K_{ij} = \exp(\widetilde{M}_{ij}/\varepsilon)$ on nonzeros, with entropic temperature $\varepsilon > 0$. We solve (entropic, unbalanced) scaling in the log-domain:

$$\log P_{ij} = \log K_{ij} + u_i + v_j, \quad (i, j) \in \text{supp}(\widetilde{M}), \quad (17)$$

$$\sum_j P_{ij} \leq \rho_i, \quad \sum_i P_{ij} \leq \gamma_j, \quad (18)$$

where $\rho_i \in (0, 1]$ and $\gamma_j \in (0, 1]$ are row/column capacity bounds. We set $\rho_i \propto w_i$ (normalized so $\rho_i \leq 1$) to reflect class weights and choose uniform $\gamma_j = \bar{\gamma}$ (e.g., $\bar{\gamma} = 1$).

Numerical stabilization and convergence: We update (u, v) by Sinkhorn–Knopp on the sparse support using log-sum-exp to prevent underflow; at iteration t :

$$u_i^{(t+1)} \leftarrow \min\left(u_i^{(t)} + \log \rho_i - \text{LSE}_j(\log K_{ij} + u_i^{(t)} + v_j^{(t)}), U_{\max}\right),$$

$$v_j^{(t+1)} \leftarrow \min\left(v_j^{(t)} + \log \gamma_j - \text{LSE}_i(\log K_{ij} + u_i^{(t+1)} + v_j^{(t)}), V_{\max}\right),$$

with caps $U_{\max}, V_{\max}$ to avoid divergence on nearly empty rows/cols. Terminate when $\Delta_{\text{KL}} < 10^{-4}$ or after $I_{\max} = 50$ iterations, or if the time budget T_{soft} (default 600 ms) is exceeded. If $|S| \cdot k > 10^6$ or the budget is exceeded, **fallback** to per-row maximum $\max_j \widetilde{M}_{ij}$ without computing P ; coverage C remains interpretable (Eq. 5).

Coverage and explanations: We use the per-row maximum $\max_j \widetilde{M}_{ij}$ for C (Eq. 5) to maintain interpretability. The top- K neighbors and (optionally) P_{ij} provide explainable neighbors for each s_i in the evidence diff.

D. Stage 2: Priority-Ordered Hard Matching

The hard matcher enforces crisp policy semantics and drives E and U (Sec. II). We greedily match highest-similarity pairs subject to class priorities:

- 1) Sort declared items s_i by decreasing *priority key* (critical $\rightarrow$ important $\rightarrow$ peripheral, $w_i, \max_j \widetilde{M}_{ij}$).
- 2) For each s_i , scan its candidate list $\mathcal{N}_k(i)$ in decreasing $\widetilde{M}_{ij}$: assign the first r_j that is unassigned and satisfies class floor $\widetilde{M}_{ij} \geq \tau_{c(s_i)}$.
- 3) Mark any unmatched s_i as missing; any unassigned r_j as extraneous (excluding ROAs).
- 4) Any matched critical artifact with invalid/unknown signature triggers unsigned status regardless of $\widetilde{M}_{ij}$.

This produces the mapping π used by E, U and the per-class sim_ℓ used in Eq. (9).

Algorithm 2 SRCV Alignment (Sparse Soft Assign $\rightarrow$ Hard Matching)

- 1: **Input:** $f_S(S)$, $f_R(R)$, class floors τ , weights w , ROA allowlist $\mathcal{A}$
 - 2: Build ANN over $\{f_R(r_j)\}$; for each s_i get $\mathcal{N}_k(i)$ and compute $\widetilde{M}_{ij}$
 - 3: If $|S| \cdot k \leq 10^6$ and within T_{soft} : run sparse log-domain Sinkhorn to get P ; else skip
 - 4: Compute coverage C via $\max_j \widetilde{M}_{ij}$; cache top- K neighbors for explanations
 - 5: Priority-sort s_i ; greedily assign pairs above $\tau_{c(s_i)}$; mark remaining s_i as missing, $r_j \notin \pi$ as extraneous ($\setminus \mathcal{A}$)
 - 6: Compute E, U, Q from Sec. II; form SRCV_{raw} (Eq. 9)
 - 7: **Output:** mapping π , coverage C , penalties (E, U, Q) , decision score SRCV_{raw}
-

E. Complexity and Memory

Let d be the embedding dimension (default 256), k the candidate fanout, and I the Sinkhorn iterations actually executed.

- **ANN queries:** $O(|S| \log |R|)$ (HNSW typical) with small constants; amortized by warm indices.
- **Sparse Sinkhorn:** $O(|S| \cdot k \cdot I)$ time and $O(|S| \cdot k)$ memory; bounded by I_{max} and T_{soft} .
- **Hard matching:** $O(|S| \cdot k)$.
- **Memory:** vectors $\approx (|S| + |R|) \cdot d \cdot 4$ bytes; sparse graph $\approx |S| \cdot k \cdot (4-8)$ bytes indices + float32 similarities.

Example envelope: $|S| = 3,000$, $|R| = 8,000$, $k = 24$, $d = 256$: sparse nonzeros $\approx 72\text{k}$; Sinkhorn with $I \leq 30$ fits in $\ll 100\text{ MB}$; stage-1+2 runtime meets the 600 ms + 400 ms budgets on current x86 nodes.

F. Numeric Guardrails and Fallbacks

We clamp similarities to $[0, 1]$, use log-sum-exp in Eq. (17), cap u, v to avoid runaway scaling on empty rows/cols, and enforce iteration/time budgets. If ANN or attestation telemetry is unavailable, the system enters *degraded mode*: coverage uses per-row maxima; policy relies on signature/attestation rules only; the evidence diff sets `degraded_mode=true`.

G. Default Parameters and SLOs

Table III lists default alignment parameters; Sec. II covers threshold calibration on SRCV_{raw} .

TABLE III

ALIGNMENT DEFAULTS (PER ENCLAVE), TUNABLE VIA CALIBRATION.

Parameter	Default	Rationale
Embedding dim d	256	accuracy/latency balance
K-NN fanout k	24	recall vs. sparsity
Entropy temp ϵ	0.05	stable soft plans
Sinkhorn I_{max}	50 iters	convergence bound
Soft budget T_{soft}	600 ms	fits 3 s SLO
Hard budget T_{hard}	400 ms	policy overhead
Row caps ρ_i	$\propto w_i$ (clipped ≤ 1)	class weight
Col caps γ_j	1.0	partial transport
Class floors τ_ℓ	learned per class	guardrails on critical

V. POLICY GATE, EVIDENCE DIFF, AND OPERATIONALIZATION

SRCV decisions are enforced by a policy-as-code gate that consumes the score in Eq. (9), class-floor checks, and operational sidecars (exceptions, overrides, ROA allowlists, degraded-mode flags). The gate returns an `allow|quarantine|block` decision, reasons, a link to an evidence diff, and (on block/quarantine) triggers forensics capture.

A. Gate Inputs and Outputs

Inputs.

- SRCV_{raw} ; per-class similarities for critical classes $\{\text{sim}_\ell\}$; findings sets `missing`, `extraneous`, `unsigned`.
- Thresholds: enclave cutoff t^* and class floors $\{\tau_\ell\}$ (calibrated per Sec. II).
- Sidecars: signed exceptions (time windows, expected deltas), override token (two-person control), ROA allowlist id/state, telemetry trust (firmware tier, attestation presence), `model_version`, `degraded_mode` flag.

Outputs.

- Decision: `allow|quarantine|block`; `reasons[]` (structured codes); link to evidence diff; forensics task id (if applicable).

B. Decision Procedure

Hard safety predicates are evaluated before score thresholds. Exceptions and ROA allowlists qualify specific findings; overrides are tightly time-boxed and audited.

Algorithm 3 SRCV Policy Gate (decision flow)

- 1: **Input:** SRCV_{raw} , $\{\text{sim}_\ell\}$, findings, t^* , $\{\tau_\ell\}$, exceptions, override, ROA, telemetry_trust, model_version, degraded_mode
 - 2: `reasons` $\leftarrow []$
 - 3: **if** degraded_mode **then** `reasons.append(degraded_mode)`
 - 4: **for** critical class ℓ **do**
 - 5: **if** $\text{sim}_\ell < \tau_\ell$ **or** findings.unsigned contains class ℓ **then**
 - 6: `reasons.append(critical_violation( $\ell$ ))`;
 - 7: **return** block, reasons
 - 8: **end if**
 - 9: **end for**
 - 10: Apply signed exceptions (within window) to downgrade expected deltas for non-critical classes; apply ROA allowlist to filter extraneous ROAs (Sec. II)
 - 11: **if** $\text{SRCV}_{\text{raw}} \geq t^*$ **then**
 - 12: **return** allow, reasons
 - 13: **else if** $t^* - \delta \leq \text{SRCV}_{\text{raw}} < t^*$ **then**
 - 14: $\{\delta \text{ is a narrow quarantine band}\}$
 - 15: **return** { quarantine, reasons }
 - 16: **else**
 - 17: `reasons.append(below_threshold)`
 - 18: **return** block, reasons
 - 19: **end if**
-

C. Rego Policy Sketch (Gatekeeper/OPA)

We provide a concise sketch that mirrors Algorithm 3. For production, thresholds and floors are sourced from a signed policy registry; exceptions and overrides are separately verified.

```
// package: srcv.gate

default allow := false
default decision := "block"
default reasons := []

critical := {"kernel", "gpu_driver", "firmware", "bmc",
            ", "libc"}

# Record degraded mode for audit, but do not allow
# it to mask violations
reasons = rs { some r in input.flags; r == "
    degraded_mode"; rs := ["degraded_mode"] }

# Hard fail on critical floor or unsigned
deny[reason] {
    some s in input.critical_sims
    s.class == critical[_]
    s.sim < input.thresholds.class_floors[s.class]
    reason := sprintf("critical_violation:%s", [s.
        class])
}
deny[reason] {
    some u in input.findings.unsigned
    u.class == critical[_]
    reason := sprintf("critical_unsigned:%s", [u.class
        ])
}

# Apply signed exceptions for non-critical classes
# within time window
exception_applies {
    some ex in input.exceptions
    ex.signature_valid == true
    input.now >= ex.valid_from
    input.now <= ex.valid_to
}

# ROA allowlist excludes approved runtime-only
# artifacts from extraneous
is_roa_ok {
    input.roa.allowlist_valid == true
    # implementation detail: engine filters extraneous
    # before policy
}

# Score threshold with small quarantine band delta
deny[reason] {
    not deny[_]
    input.srcv_raw < input.thresholds.global
    reason := sprintf("below_threshold:%.3f<%.3f", [
        input.srcv_raw, input.thresholds.global])
}

allow {
    not deny[_]
}
decision := "allow" { allow }
decision := "quarantine" {
    not deny[_]
    input.srcv_raw >= input.thresholds.global - input.
        thresholds.delta
    input.srcv_raw < input.thresholds.global
}
```

D. Evidence Diff Payload

The evidence diff links decisions to human-auditable facts: global score and terms (C, E, U, Q); matched pairs and per-class similarities; reason codes; attention weights; model versioning; and operational flags.

TABLE IV
EVIDENCE DIFF: REQUIRED TOP-LEVEL FIELDS.

Field	Description
node_id, evaluated_at	Host identifier; timestamp (UTC)
model_version(s)	Active (and prior, if dual-embedding forensics mode)
global	$\text{SRCV}_{\text{raw}}, C, E, U, Q, t^*$, decision
critical_sims	Per-class sim_ℓ and floors τ_ℓ
components[]	Declared/observed pairs, similarity, status, reason_codes[]
attention_weights	Channel weights for explanation
flags	{degraded_mode, exception_ids, roa_id}

Compact example.

```
// Truncated example; actual schema includes full
// component diffs.
{
    "node_id": "jwst-node-073",
    "evaluated_at": "2025-09-25T06:12:03Z",
    "model_version": "srcv-1.3.2",
    "global": {
        "srcv_raw": 0.964,
        "coverage": 0.992,
        "penalties": {"extraneous": 0.012, "unsigned":
            0, "anchor_gaps": 0},
        "threshold": 0.970,
        "decision": "quarantine"
    },
    "critical_sims": [
        {"class": "kernel", "sim": 0.998, "floor":
            0.990},
        {"class": "gpu_driver", "sim": 0.972, "floor":
            0.970},
        {"class": "libc", "sim": 0.969, "floor": 0.970,
            "reason_codes": ["shadow_risk"]}
    ],
    "attention_weights": {"txt": 0.18, "ver": 0.12, "
        sig": 0.34, "hash": 0.21, "path": 0.08, "lin":
        0.07},
    "flags": ["degraded_mode"],
    "components": [
        {
            "class": "gpu_driver",
            "declared": {"vendor": "NVIDIA", "version":
                "535.146.02", "signer": "NVIDIA-Root"},
            "observed": {"vendor": "NVIDIA", "version":
                "535.146.02", "signer": "NVIDIA-Root"},
            "similarity": 0.972,
            "status": "matched",
            "reason_codes": []
        },
        {
            "class": "libc",
            "declared": {"product": "glibc", "version":
                "2.35"},
            "observed": {"product": "glibc", "version":
                "2.35-custom"},
            "similarity": 0.969,
            "status": "mismatch",
            "reason_codes": ["overlay_shadow_lib"]
        }
    ]
}
```

E. Operational Considerations

Overrides. Emergency overrides are 15 min, two-person control; extensions require a third approver and SRE lead page. Same approver pair is barred for 24 h; per-approver cooldown 72 h; chained attempts raise an anomaly alert.

Forensics. On block or quarantine, the gate triggers capture (proc tree, mount table, overlay diff, selected file hashes, journal slice, Redfish snapshot); artifacts are compressed and retained per enclave policy.

SLOs. Gate evaluation must add < 50 ms to the end-to-end budget; evidence diff generation happens asynchronously but is available within 60 s for auditor workflows.

Mixed-model rollouts. During dual-embedding forensics windows, the diff includes both model versions; gating uses only the active model and never mixes models within a decision chain.

VI. CALIBRATION, SLOs, AND EVALUATION

This section describes how thresholds and class floors are learned, how probabilities and confidence intervals are produced for operator UIs, and how SRCV is evaluated against JWST enclave requirements, including red-team injects, mimicry tests, and cross-enclave generalization.

A. Calibration Dataset, Labels, and Splits

We construct a calibration corpus per enclave from: (i) golden deployments and signed maintenance windows (positives), (ii) routine operations (unlabeled), (iii) red-team and synthetic drift injects (negatives). Ambiguous events are assigned to *quarantine*.

Let $\mathcal{D}_{\text{dev}}$ and $\mathcal{D}_{\text{val}}$ be stratified splits by day and workload, preserving class mix. Metrics report means and 95% CIs via bootstrap on $\mathcal{D}_{\text{val}}$.

B. Operating-Point Selection

We optimize class floors $\{\tau_\ell\}$ and the enclave cutoff t^* on SRCV_{raw} (Eq. 9) by minimizing a class-aware cost:

$$\mathcal{J}(\tau, t^*) = \sum_{\ell \in \mathcal{C}} [c_{\text{fn}}(\ell) \text{FNR}_\ell + c_{\text{fp}}(\ell) \text{FPR}_\ell], \quad (19)$$

with $c_{\text{fn}}(\ell) \gg c_{\text{fp}}(\ell)$ for $\ell \in \mathcal{C}_{\text{crit}}$. We impose *guardrails*:

$$\text{FNR}_\ell \leq 0.001 \quad \forall \ell \in \mathcal{C}_{\text{crit}} \quad (95\% \text{ CI}). \quad (20)$$

The quarantine band δ is selected to capture the highest-uncertainty region around t^* while capping operator load:

$$\delta = \arg \min_{\delta \geq 0} \text{OpsCost}(\delta) \quad \text{s.t.} \quad \Pr[t^* - \delta \leq \text{SRCV}_{\text{raw}} < t^*] \leq \beta \quad (21)$$

with β the maximum quarantine fraction (default $\beta=5\%$).

Algorithm 4 Threshold Calibration (per enclave)

- 1: Assemble $\mathcal{D}_{\text{dev}}, \mathcal{D}_{\text{val}}$; compute (C, E, U, Q) and SRCV_{raw}
- 2: Grid or Bayesian search over $\{\tau_\ell\}$ (critical first), then t^* to minimize $\mathcal{J}$ (Eq. 19)
- 3: Enforce guardrails $\text{FNR}_\ell \leq 0.1\%$ (95% CI) for all critical classes
- 4: Select quarantine band δ per operator-load constraint β
- 5: Bootstrap 95% CIs for $\{\tau_\ell, t^*\}$; write signed policy bundle to registry

C. Probability Calibration and Confidence Intervals

For UI display only, we fit a monotone isotonic map $\hat{P}(\text{allow} \mid \text{SRCV}_{\text{raw}})$ on $\mathcal{D}_{\text{dev}}$, then validate Expected Calibration Error (ECE) on $\mathcal{D}_{\text{val}}$. We precompute 95% bootstrap CIs per score bucket to avoid runtime cost; ties are handled by local linear interpolation within the bucket.

D. Performance SLOs and Monitoring

We instrument end-to-end latency as:

$$T_{\text{total}} = T_{\text{extract}} + T_{\text{ANN}} + T_{\text{soft}} + T_{\text{hard}} + T_{\text{policy}}.$$

Targets: provisioning $p95 \leq 3$ s, container admission $p95 \leq 500$ ms; policy evaluation ≤ 50 ms (Sec. V). We track: (i) decision latency $p50/p95$, (ii) fraction of degraded_mode decisions ($\leq 2\%$ /day), (iii) override usage and chaining alerts, (iv) failure budget consumption (critical false allows/quarter).

E. Evaluation Protocols

Ablations. Single-tower vs. dual-head; attention on/off; channel dropout on/off; with/without E_{state} . **Success:** dual-head $\geq 2\%$ AUROC gain over single-tower (paired t -test, $p < 0.05$), no SLO regression.

Mimicry and Adversarial Robustness. PGD text/version adversaries with $L_\infty \epsilon \in \{10^{-3}, 10^{-2}, 10^{-1}\}$ under vendor/semver regex constraints; structural adversaries with signer-nullification and lineage family swaps preserving anchor conflicts. **Success:** $\geq 99.9\%$ block for critical anchor-conflict cases.

Cross-Enclave Generalization. Train on Enclave-A, test on Enclave-B with vendor shifts. Trigger transfer learning when MMD/PSI drift exceeds $\mu + 3\sigma$ of a 30-day baseline. **Success:** AUROC drop $\leq 2\%$; if exceeded, post-transfer recovers $\geq 80\%$ of lost AUROC.

Operator Study. Time-to-resolution with evidence diff; **Success:** median ≤ 5 min; inter-rater agreement $\kappa \geq 0.7$ on reason codes.

F. Drift and Stability Metrics

We monitor: (i) cosine angle drift for stable components across model versions, (ii) attention entropy floors to detect collapse, (iii) embedding distribution drift via Maximum Mean Discrepancy (MMD) and Population Stability Index (PSI). Drift alarms initiate shadow evaluation and, if persistent, transfer learning (Sec. III).

G. Red-Team Exercises and Coverage

Quarterly planted violations emulate unsigned critical modules, overlay shadowing of `libc`, and firmware family mismatches. *Success*: detect $\geq 95\%$ of injected *critical* violations; report dwell time, override usage, and any false blocks.

H. Reporting and Policy Publication

Calibration artifacts (thresholds $\{\tau_\ell\}$, cutoff t^* , quarantine band δ), validation metrics, and CIs are bundled, signed, and published to the policy registry consumed by the gate (Sec. V). Dashboards display SRCV_{raw} , bucketed $\hat{P}(\text{allow})$ with CIs, reason codes, and SLO compliance.

TABLE V
EVALUATION METRICS AND TARGET THRESHOLDS (PER ENCLAVE).

Category	Metric	Target
Detection	FNR (critical classes)	$\leq 0.1\%$ (95% CI)
Calibration	ECE (display probability)	$\leq 2\%$ absolute
Latency	Provisioning decision p95	≤ 3 s
Robustness	Anchor-conflict block rate	$\geq 99.9\%$
Ops	Quarantine median review time	≤ 5 min
Ops	Degraded decisions fraction	$\leq 2\%/day$
Generalization	Cross-enclave AUROC drop	$\leq 2\%$ (pre-transfer)

VII. ADVERSARIAL ROBUSTNESS, FIRMWARE TRUST, AND LIMITATIONS

This section specifies the threat model, the concrete defenses implemented by SRCV, the operational realities of firmware telemetry with graded enforcement, privacy boundaries for cross-enclave signals, and explicit limitations.

A. Adversary Model

We assume an adversary who can: (i) introduce runtime artifacts (modules, overlays, layers) with misleading vendor/version strings; (ii) tamper with string-reported firmware fields (e.g., Redfish version, family); (iii) attempt to game similarity by imitating high-similarity neighborhoods; but *cannot* forge cryptographic anchors (hashes bound to measured boot, trusted signer roots) or TPM quotes. Network and host collectors are assumed authenticated; collector compromise is out of scope for Phase 1.

B. Mimicry and Similarity-Gaming Defenses

SRCV hardens against mimicry at training and inference time:

- **Adversarial training.** PGD-crafted text/version negatives (Sec. III) under vendor/semver regex constraints, plus structural negatives that preserve anchor conflicts (signer-nullification, lineage swaps).
- **Cryptographic anchor consistency.** For critical classes, a pass requires agreement between similarity and anchors; otherwise the class-floor indicator in Eq. (9) fires regardless of C .
- **Temperature-scaled similarity.** We apply temperature scaling on cosine scores for critical classes to reduce overconfidence in tightly clustered neighborhoods.

- **Attention safeguards.** Early-epoch entropy regularization prevents single-channel dominance; at inference, low entropy triggers uniform-attention fallback and flags `attention_collapse` in the evidence diff (Sec. III).

C. Telemetry Tamper and Degraded Mode

SRCV cross-checks kernel-reported state with IMA/TPM evidence. When attestation is absent or stale, the decision proceeds in *degraded mode*: (i) C uses per-row maxima without soft assignment, (ii) gating falls back to signature and class floors, (iii) the evidence diff sets `degraded_mode=true` for audit (Sec. V). Sustained degraded operation ($> 2\%$ of daily decisions) triggers post-mortem and remediation (Sec. VI).

D. Firmware Trust Tiers and Enforcement

Firmware telemetry is heterogeneous across vendors; SRCV enforces graded policy via trust tiers.

TABLE VI
FIRMWARE TRUST TIERS AND ENFORCEMENT POLICY.

Tier	Telemetry Basis	Enforcement
T0	Measured boot/TPM quotes, IMA/EVM	Eligible for hard gating
T1	Vendor-signed + consistent Redfish IDs	Observe $\rightarrow$ hard gate once stable
T2	String-only reporting (no verifiable chain)	Observe-only; high q_{fw} penalty

SRCV begins with T1/T2 in observe-only mode; T1 can graduate to enforcement after a 30-day stability period with `redfish_consistency_score` ≥ 0.9 and no contradictory anchors (Sec. V).

E. Runtime-Only Artifacts (ROA) Governance

Legitimate runtime-born objects (JIT code, ephemeral at-test tokens, cache stubs) are controlled by a signed, time-boxed ROA allowlist (Sec. II). Expired allowlists produce quarantine with a 24h grace and ticketed auto-renew for critical workloads; suspicious ROA patterns (e.g., ELF in unexpected ROA path) escalate regardless of allowlist.

F. Cross-Enclave Signals and Privacy

To detect wide-area anomalies without crossing enclave boundaries, SRCV exchanges *privacy-preserving sketches* only: MinHash/SimHash summaries of `reason_code` distributions and component-family IDs (no paths, no usernames, no raw hashes). Sketches are signed, time-bounded, and rotated; correlation alerts prompt local investigation without sharing raw runtime data (Sec. VI).

G. Residual Limitations

Despite defenses, several limitations remain:

- **Label scarcity and bias.** Golden deployments and signed maintenance windows do not cover the full tail of benign variability; active-learning caps can slow convergence (Sec. VI).
- **ROA policy drift.** Overly broad allowlists can mask malicious artifacts; overly narrow lists increase quarantine

load. We mitigate via time-boxing, review, and red-team injects but residual risk remains.

- **Firmware opacity.** In T2 environments, string-only firmware reporting can create false complacency; SRCV addresses via penalties (q_{fw}) and observe-only policy but cannot create trust evidence that vendors do not expose.
- **Collector dependency.** SRCV assumes trustworthy collection; if a host or BMC collector is compromised, anchors may be selectively hidden. Long-term mitigation requires independent attestation paths.
- **Compute budget coupling.** The 3 s provisioning SLO bounds soft matching and ANN; extremely large $|S|$ or $|R|$ can force frequent fallbacks, reducing explanatory richness for operators.
- **Interpretability granularity.** While attention weights and reason codes aid audit, per-token contributions within E_{txt} are not exposed to avoid overfitting explanations; this is a trade-off against potential feature leakage.

H. Open Problems and Future Hardening

- **Third-party firmware attestation.** Standardizing vendor-neutral, remotely verifiable firmware evidence for BMC/UEFI remains open; SRCV’s tiering can incorporate such signals when available.
- **Causal triage.** Linking evidence diffs to causal explanations (e.g., which config step induced the drift) would reduce operator time-to-resolution; requires provenance hooks in provisioning.
- **Joint model-policy optimization.** Co-optimizing $\{\tau_\ell, t^*\}$ with human workload cost models could better balance quarantine load vs. risk under shift.
- **Hardened collectors.** Securing scrapers with remote attestation and redundant channels (e.g., BMC-side attest + OS-side attest) reduces single-point tamper risks.

VIII. CONCLUSION AND OUTLOOK

SRCV reframes SBOM verification for NASA JWST science platforms as a *semantic* concordance problem between a signed golden image and live runtime state. By coupling a shared-bottom, dual-head metric model with a two-stage (soft→hard) matcher and an interpretable linear decision score, SRCV provides graded assurance, clear operator actions, and auditor-ready evidence diffs—while meeting strict provisioning SLOs. The design integrates firmware trust tiers, runtime-only artifact (ROA) governance, active-learning safeguards, and cross-enclave, privacy-preserving outbreak detection.

Operationally, SRCV enables *policy-as-code gating* from BIOS/UEFI through container admission, turning “IaC with embedded security” into measurable, explainable enforcement. The evaluation plan balances detection targets (e.g., $\geq 99.9\%$ block of unsigned/unknown *critical* components) with operator load (quarantine band, 5 min median review) and generalization under vendor drift.

Future work includes: (i) vendor-neutral, remotely verifiable firmware attestations that allow T1→T0 graduation across more platforms; (ii) provenance hooks to tie evidence diffs to

causal change steps; and (iii) joint optimization of thresholds with human cost models to minimize alert fatigue under distribution shift. As JWST infrastructures evolve, SRCV’s modular architecture (encoders, matcher, gate) supports incremental hardening without disrupting science operations.