Anchor: A Federated Governance Engine for Secure and Compliant Agentic AI Systems

Tanishq Dasari
Independent Researcher, AnimusLab, India
tan@anchorgovernance.tech
github.com/AnimusLab/anchor │ anchorgovernance.tech
Submitted: April 2026 · Preprint: Zenodo

Abstract

As agentic AI systems transition from closed-loop sandboxes to open-world execution environments, the risk of unaligned tool use, data leakage, and regulatory non-compliance increases substantially. Existing governance tools address either static code quality or narrow runtime monitoring, but none provide a unified, cryptographically verifiable enforcement layer that spans the full deployment lifecycle of an AI system.

We present Anchor, a federated governance engine that enforces multi-lens compliance across heterogeneous agentic architectures through two complementary mechanisms. Layer 1 (Anchor v1–v4) performs static analysis of AI-adjacent source code using Tree-sitter abstract syntax tree (AST) parsing against a constitution.anchor rule set — a signed Universal Constitution defining what is permitted — and a mitigation.anchor detection catalog defining how violations are identified. Layer 2 (AnchorRuntime, v4.3.5) intercepts live AI inference calls, hashes inputs and outputs with SHA-256, evaluates outputs against the same constitution at runtime, and writes each decision into an HMAC-signed, hash-chained append-only audit log: the Decision Audit Chain (DAC).

Central to Layer 1 enforcement is the Diamond Cage, a WebAssembly (WASM)-based behavioral verification sandbox built on WasmEdge. Governance integrity is maintained through a remote lockfile model (GOVERNANCE.lock) sealing 18 policy files via SHA-256. Layer 2 introduces a dual-mode enforcement model — Structured Mode for high-stakes decisions (mandatory JSON with ReasonCode and FeatureAttribution) and Conversational Mode for general interactions — alongside a deterministic ETH domain that replaces probabilistic bias classifiers with Governance Invariants: an Aho-Corasick trie scanning structured attribution fields against a lintable prohibited-proxy taxonomy. Regulatory output methods (cims_payload(), adverse_action_reasons(),

eu_article12_record()) translate a single AuditEntry into jurisdiction-specific dialects without separate implementations.

We validate the system against four open-source codebases: FINOS Architecture-as-Code, HuggingFace Hub, Django, and OpenSpiel. We demonstrate that a single constitution.anchor file simultaneously satisfies Article 12 logging requirements of the EU AI Act (2024/1689), RBI FREE-AI Report Recommendations 7 and 14, CFPB Regulation B adverse action obligations, SEC 2026 Examination Priorities, and the NIST AI Risk Management Framework — a property we term regulatory polyglottism.

Anchor v4.3.5 is available at github.com/AnimusLab/anchor and on PyPI as anchor-audit.

Keywords: AI governance; agentic AI; federated policy; audit trail; EU AI Act; WebAssembly; cryptographic logging; static analysis; Decision Audit Chain; fintech compliance; ETH governance invariants; runtime interception

1. Introduction

The emergence of agentic AI systems — large language models equipped with tool-calling capabilities, persistent memory, and multi-step planning — has created a new class of governance challenge. Unlike traditional software, an agentic system does not execute a fixed program; it dynamically decides which tools to invoke, which data to access, and which actions to take based on model outputs that are inherently non-deterministic. The attack surface is correspondingly large: prompt injection, credential harvesting via environment variable access, unsandboxed subprocess execution, unvalidated LLM output consumption, and data poisoning of retrieval stores are all patterns that existing static analysis tools were not designed to detect.

This challenge is compounded by a convergent global regulatory response. The EU AI Act (2024/1689) classifies credit scoring, AML monitoring, and fraud detection as high-risk AI systems subject to mandatory conformity assessments, logging, and human oversight documentation [1]. The Reserve Bank of India's FREE-AI Report [2] issued 26 mandatory recommendations for AI in financial services, including per-decision audit trails reportable to the CIMS portal. The U.S. Consumer Financial Protection Bureau [3] established through a $45 million enforcement action against Goldman Sachs that AI-assisted credit decisions must be explainable at the decision level. The SEC identified AI governance as the top examination priority for 2026 [5], and the NIST AI RMF [6] establishes governance, measurement, and management functions as baseline requirements.

Against this backdrop, we identify a dual governance gap:

- Pre-deployment gap — no tooling verifies AI-adjacent source code against a signed constitutional rule set before it ships.
- Runtime gap — no unified, cryptographically verifiable record proves what an AI system decided, when, on what input, under which rules, with what ethical outcome.

Anchor closes both gaps through a single signed constitution.anchor that governs both layers. This paper makes the following contributions:

- We introduce the Anchor governance engine, spanning static code analysis (v1–v4) and runtime decision auditability (AnchorRuntime v4.3.5), unified by a federated constitutional model.
- We describe the Diamond Cage, a WASM-based behavioral verification sandbox that proves security invariants through execution rather than pattern matching.
- We define the Decision Audit Chain (DAC), a hash-chained, HMAC-signed append-only log satisfying the logging requirements of five major regulatory frameworks from a single data structure.
- We introduce GOVERNANCE.lock, a remote integrity lockfile mechanism sealing 18 policy files, preventing tampering with local governance rules.
- We present a violation taxonomy of 47 types across nine domain namespaces, validated against four open-source codebases.
- We introduce the Governance Invariant model for ETH domain enforcement: a deterministic, classifier-free approach to bias detection using an Aho-Corasick trie against a lintable prohibited-proxy taxonomy.
- We introduce the dual-mode enforcement model (Structured vs. Conversational) with the @anchor.enforce decorator API.
- We demonstrate regulatory polyglottism: a single AuditEntry simultaneously satisfying five jurisdictional frameworks via polymorphic dialect output methods.

2. Background and Related Work

2.1 Regulatory Landscape

EU AI Act (2024/1689). Articles 9–13 impose mandatory requirements on high-risk AI systems. Article 12 requires automatically generated, tamper-evident logs retained per operator specification [1]. Full enforcement on high-risk systems begins August 2026, with fines up to €30 million or 6% of global annual revenue.

RBI FREE-AI Report (2025). Recommendations 7 and 14 mandate board-approved AI policies, per-decision audit trails reportable to the CIMS portal, and explainable credit decisions [2]. Recommendation 19 requires mandatory algorithmic fairness audits for AI systems used in credit and lending.

CFPB Regulation B. The 2024 guidance [4] extended adverse action notice obligations to AI-assisted credit decisions. The $45 million Goldman Sachs enforcement [3] established that decision-level reason codes must be producible on demand — a requirement Anchor satisfies through the ETH-002 No-Prose Rule and adverse_action_reasons() method.

NIST AI RMF 1.0. The Govern, Map, Measure, and Manage functions require documented policies, accountability structures, and audit trails of AI system behaviour [6].

FCA and SEBI. The FCA's 2024 AI governance guidance [7] requires human oversight records and model version traceability. SEBI's algorithmic trading circular [8] requires order-level traceability for trading AI.

2.2 Related Technical Work

Static analysis for software compliance has a long history [24]. Existing tools — Bandit [25], Semgrep [26], SonarQube — address general software vulnerabilities rather than AI-specific constitutional governance. NeMo Guardrails [16] and Guardrails.ai focus on prompt and output validation but do not provide code-level governance or cryptographic audit trails. Open Policy Agent [17] provides policy-as-code for general systems; Anchor specialises this to the risk taxonomy of agentic AI.

Schneier and Kelsey [14] established foundational principles for secure audit logs. Crosby and Wallach [15] formalised tamper-evident data structures. Kao [9] proved evidence binding, tamper detection, and non-equivocation for constant-size cryptographic evidence structures — a construction our DAC instantiates in the financial compliance domain.

Prior bias detection approaches in AI governance rely on probabilistic classifiers, creating a category error for regulatory contexts: regulators require logical proof, not confidence scores. Anchor's Governance Invariant model (Section 4.4) addresses this gap deterministically.

To our knowledge, no prior work unifies pre-deployment constitutional governance with runtime decision auditability under a single cryptographically signed constitution, provides WASM-based behavioral verification, implements classifier-free ETH enforcement via Governance Invariants, or demonstrates regulatory polyglottism across five jurisdictional frameworks from a single audit data structure.

3. System Architecture

3.1 Design Principles

Anchor is designed around five principles:

- Single constitution, dual enforcement. One signed rule set governs both static code analysis and runtime output evaluation.
- What vs. How separation. The constitution.anchor defines what is permitted (semantic laws). The mitigation.anchor defines how violations are detected (AST queries, context-aware regex, Aho-Corasick taxonomy). These are independently versioned and sealed.
- Governance floor invariant. Open Laws define the minimum floor. Private policy.anchor can only raise severity, never lower it.
- Cryptographic non-repudiation. Every governance artefact carries a SHA-256 hash and where applicable an HMAC-SHA256 signature.
- Regulatory polyglottism. A single AuditEntry simultaneously satisfies the logging requirements of multiple jurisdictions without separate implementations per regulator.

3.2 Overall Architecture

Anchor operates on two layers unified by a single signed constitution. Layer 1 operates pre-deployment on source code; Layer 2 operates at runtime on live AI inference.

Rule Consolidation and Scan Efficiency. Anchor v4.3.5 loads 170+ active laws across 9 domains, 3 frameworks, and 6 regulatory standards. Rule Consolidation maps multiple compliance IDs to

a single underlying detection pattern. The shell injection pattern fires once and simultaneously satisfies [FINOS-014, OWASP-002, RBI-018, SEC-007] — four compliance references from one scan pass. This maintains linear scan time complexity regardless of the number of active regulatory frameworks.

```
[SOURCE CODE]
  ↓ Tree-sitter AST parse
[Anchor v1-v4 Static Analyser]
  ↓ constitution.anchor + mitigation.anchor
  ↓ Diamond Cage (WASM behavioral verify)
[ViolationLog + GOVERNANCE.lock check]
  ↓
[AI INFERENCE CALL]
  ↓ @anchor.enforce(mode, jurisdiction)
[AnchorRuntime Wrapper]
  SHA256(input) ‖ call model ‖ SHA256(output)
  ↓ evaluate against constitution.anchor
  ↓ ETH Governance Invariant check
[Decision Audit Chain (DAC)]
  HMAC-signed ‖ hash-chained ‖ append-only
  ↓
[/audit API] ← regulator query
```

Figure 1: Anchor two-layer governance architecture with ETH Governance Invariant layer.

3.3 Federated Domain Architecture

Anchor V4 transitions from a monolithic rule catalog to a federated domain model organised as a governance/ directory with three subdirectories:

Domains (domains/) are always loaded and form the constitutional floor across nine namespaces: SEC (Security), ETH (Ethics), PRV (Privacy), ALN (Alignment), AGT (Agentic AI), LEG (Legal), OPS (Operational), SUP (Supply Chain), and SHR (Shared cross-domain).

Frameworks (frameworks/) are opt-in mappings to standards bodies: FINOS AI Governance Framework (23 risks), OWASP LLM Top 10 (2025), NIST AI RMF, and ISO 42001.

Regulators (government/) are opt-in jurisdictional files: RBI FREE-AI 2025, EU AI Act 2024/1689, CFPB Regulation B, FCA 2024, SEBI AI/ML, and SEC 2026 Examination Priorities.

GOVERNANCE.lock seals all 18 policy files with SHA-256 hashes, verified against the authoritative AnimusLab registry on every anchor check run.

3.4 The What vs. How Separation

Constitutional Manifest (constitution.anchor)

Defines semantic laws in a typed, semantically versioned rule format. Each rule carries severity, min_severity (the governance floor), maps_to (multi-jurisdiction references), and where

applicable runtime_patterns (regex for Layer 2 enforcement) or prohibited_proxies (ETH-001 taxonomy).

Mitigation Catalog (mitigation.anchor)

Defines how violations are detected. Tree-sitter AST queries provide language-aware structural matching for Layer 1. Context-aware regex patterns are scoped to their semantic risk. Runtime patterns in domain files extend enforcement to Layer 2 without duplicating rule definitions.

3.5 Alias Resolution and Multi-ID Reporting

Anchor maintains backward compatibility through a Rosetta Stone alias registry. The loader resolves alias chains while inheriting the highest severity floor encountered. In scan output, findings are reported with the full many-to-many resolution chain:

```
[FINOS-014, RBI-018, SEC-007] Shell Injection
Location: src/agent.py:47
Statute: FINOS Ri-014, RBI FREE-AI Cybersecurity, SEC-007
```

3.6 Diamond Cage — WASM Behavioral Verification

The Diamond Cage is Anchor's behavioral enforcement mechanism built on WasmEdge [18]. Where static analysis detects dangerous patterns, the Diamond Cage proves dangerous behaviour through execution. When an agent attempts to invoke a tool T with arguments A: (1) the call is intercepted by the Anchor runtime layer; (2) parameters (T, A) are cross-referenced with active constitutional laws; (3) if a violation is detected, the runtime blocks the call or applies safe truncation. Diamond Cage overhead is < 5 ms per tool invocation.

3.7 GOVERNANCE.lock — Remote Integrity Verification

GOVERNANCE.lock seals 18 operational files across domains/, frameworks/, and government/ via SHA-256. Any modification — including formatting changes — produces a hash mismatch. Under seal_check: strict, the scan aborts with a detailed integrity violation report. The anchor sync --restore command fetches authoritative governance files from the registry and restores tampered files, logging all restore operations to a hash-chained sync.log.

4. Layer 2: AnchorRuntime and the Decision Audit Chain

4.1 Interception Model

AnchorRuntime operates as a transparent wrapper around AI model inference via the @anchor.enforce decorator. The decorator introduces a dual-mode enforcement model:

- Structured Mode (mode="structured") — for high-stakes decisions (credit, medical, trading). The AI is required to produce structured JSON output. Prose output is itself a BLOCKER violation (ETH-002). Prohibited proxy scanning runs only on typed attribution fields.

- Conversational Mode (mode="conversational") — for customer support, internal assistants, general chat. Aho-Corasick taxonomy runs against full response text.

```
@anchor.enforce(mode="structured", jurisdiction="RBI")
def approve_loan(applicant_data):
    return llm.call(prompt_template, applicant_data)

@anchor.enforce(mode="conversational", domains=["PRV", "ETH"])
def support_chatbot(user_query):
    return llm.chat(user_query)
```

The mode declaration becomes part of the audit trail. A regulator can verify not only what the AI decided but what governance guarantees were active for that decision class.

4.2 AuditEntry Structure

Each intercepted inference produces an AuditEntry with the following canonical schema:

```
@dataclass
class AuditEntry:
    entry_id        : str   # UUID
    timestamp       : str   # ISO8601 UTC
    provider        : str   # model provider
    jurisdiction    : str   # active regulatory context
    project_name    : str
    git_commit      : str
    is_compliant    : bool
    status          : str   # CLEAN | VIOLATION
    findings_hash   : str   # SHA256(sorted rule_ids)
    prev_chain_hash : str
    chain_hash      : str   # DAC chain link
    signature       : str   # HMAC-SHA256
    violations      : list  # Violation dicts
    telemetry       : dict
```

4.3 Decision Audit Chain (DAC)

The DAC is an append-only, hash-chained log of AuditEntry records persisted to .anchor/runtime_chain.jsonl. Each entry's chain hash is computed as:

$$H_n = \text{SHA256}(H_{n-1} \parallel e_n.\text{findings\_hash})$$

where $H_0 = 0^{64}$ (genesis hash). Modification of any entry invalidates all subsequent hashes, making tampering detectable by chain verification. Each entry additionally carries an HMAC-SHA256 per-entry signature over canonical JSON. This dual-layer construction satisfies the evidence binding, tamper detection, and non-equivocation properties formalised by Kao [9].

The Zero-Knowledge strategy separates the full audit record from public telemetry. The Local Safe (.anchor/runtime_chain.jsonl) contains the complete AuditEntry including prompt and response previews. The Global Whisper to the public ledger contains only the chain_hash and signature — cryptographic proof that an audit happened with no recoverable content. Dictionary attacks against the public ledger are structurally impossible because the findings_hash is computed over sorted rule_id metadata, not response text.

4.4 ETH Domain — Governance Invariant Model

Prior approaches to runtime bias detection rely on probabilistic classifiers, creating a fundamental incompatibility with regulatory requirements: regulators require logical proof, not confidence scores. Anchor v4.3.5 introduces the Governance Invariant model for the ETH domain, replacing classifiers with two deterministic mechanisms.

ETH-002: The No-Prose Rule

In Structured Mode, the DecisionAuditor enforces a structural contract: an AI decision is not a valid decision unless it provides its mathematical homework. If the LLM returns prose instead of JSON with ReasonCode and FeatureAttribution, this is a BLOCKER violation by construction — absence of explainability is structurally impossible to miss, not probabilistically estimated. This directly addresses the Goldman Sachs / CFPB case: the violation was not a biased decision but an unexplainable one.

ETH-001: Aho-Corasick Prohibited Proxy Taxonomy

For bias detection, Anchor uses a deterministic taxonomy of prohibited proxy concepts stored in ethics.anchor as a structured concept graph. At DecisionAuditor boot, the taxonomy is compiled into an Aho-Corasick trie in RAM. In Structured Mode, the trie scans only the typed attribution fields (ReasonCode, FeatureAttribution, DenialReason, RiskFactors) — not the full response. In Conversational Mode, the full response text is scanned.

Each prohibited proxy node carries a concept name, canonical identifier, surface terms, synonym coverage status, and legal rationale. The synonym_coverage: partial field makes the taxonomy self-auditing: the Anchor linter flags any concept with incomplete coverage, and governance files with fragile taxonomies cannot be deployed to the ledger. This transforms the hand-authored fragility problem from a silent gap into a lint error.

Mechanism	Rule	Trigger	Guarantee
No-Prose Rule	ETH-002	Prose returned in structured mode	BLOCKER — 100% deterministic
Missing Fields	ETH-002	ReasonCode or FeatureAttribution absent	BLOCKER — 100% deterministic
Prohibited Proxy	ETH-001	Surface term in attribution fields	BLOCKER — trie match
Conversational Scan	ETH-001	Surface term in full response	Coverage = taxonomy completeness

Table 1: ETH Governance Invariant enforcement mechanisms.

4.5 Regulatory Output Methods

The AuditEntry exposes jurisdiction-specific output methods, enabling a single audit record to satisfy multiple regulatory frameworks without separate implementations:

```
# CFPB Regulation B adverse action
entry.adverse_action_reasons()
# -> ["Credit denied: debt-to-income ratio (V-CREDIT-001)"]

# RBI CIMS-format payload
entry.cims_payload()
# -> {"report_type": "AI_DECISION_AUDIT", "sutra": 4, ...}

# EU AI Act Article 12 log record
entry.eu_article12_record()
# -> {"log_type": "HIGH_RISK_AI_DECISION", ...}

# SEC Item 1.05 materiality disclosure
entry.sec_item105_record()
# -> {"disclosure_type": "MATERIAL_AI_RISK", ...}
```

The Polymorphic Dialect Factory ensures that a single violation — e.g., ETH-001 Redlining proxy detected — lands as a Sutra 4 (Fairness) alert for the Mumbai NOC, an Item 1.05 Materiality Disclosure for the NYC compliance team, and an Article 12 event log for EU regulators, each signed with regional HMAC keys via the Fernet-encrypted credential vault.

5. Violation Taxonomy

We define a violation taxonomy of 47 types across nine domain namespaces, mapped to regulatory obligations. The ETH domain now includes structured Governance Invariant rules in addition to pattern-based detection.

Category	Severity	Detection Method	Regulatory Mapping
Bias / Proxy (ETH-001)	BLOCKER	Aho-Corasick trie	ECOA, FHA-805, RBI-019
Explainability (ETH-002)	BLOCKER	Structural — No-Prose Rule	CFPB Reg B, RBI-014, EU Art 13
Oversight Removal (ETH-003)	BLOCKER	Runtime regex	FINOS Ri-020, FCA 2024
Toxic Output (ETH-004)	ERROR	Runtime regex	FINOS Ri-023

Category	Severity	Detection Method	Regulatory Mapping
Credential Harvest (SEC-004)	CRITICAL	AST + runtime regex	NIST AI RMF, RBI-018
Prompt Injection (SEC-001)	CRITICAL	AST + runtime regex	OWASP LLM01, FINOS Ri-001
Shell Injection (SEC-007)	CRITICAL	AST + runtime regex	FINOS Ri-014, SEC 2026
Data Poisoning (SEC-002)	CRITICAL	AST query	FINOS Ri-002, RBI Rec 19
PII Leakage (PRV-001)	CRITICAL	AST + Aho-Corasick	GDPR, DPDP 2023, RBI
Model Overreach (AGT-001)	ERROR	AST query	FINOS Ri-018, FCA 2024

Table 2: Violation taxonomy (partial). ETH domain now uses Governance Invariant detection.

6. Case Studies

We evaluate Anchor v4.3.5 against four open-source codebases of direct relevance to AI governance. All findings are reproducible by running anchor check <target> with the configuration files in case_studies/ of the repository.

6.1 FINOS Architecture-as-Code

Scan statistics: 684 files scanned, 865 ignored, 1,549 total files across 415 directories. Findings: 11 blockers, 0 warnings. All 11 blockers are [FINOS-002, OWASP-003, SEC-002] Data Poisoning: unencrypted MongoDB upsert operations across six store files. The multi-ID header demonstrates regulatory polyglottism — a single detection simultaneously satisfies FINOS-002, OWASP-003, and SEC-002.

6.2 HuggingFace Hub

Scan statistics: 164 files scanned, 54 ignored, 218 total files across 22 directories. Findings: 12 blockers, 0 warnings across three risk categories.

- Shell Injection (6 blockers): [FINOS-014, OWASP-002, RBI-018, SEC-007] across cli/lfs.py and utils/ files.
- Hallucination / LLM Output Without Validation (5 blockers): [ALN-001, FINOS-008, OWASP-009] across inference/_client.py. Each instance calls chat.completions.create() without schema validation.
- Raw Network Access (1 blocker): [FCA-004, FINOS-013, SEC-006] — direct OpenAI API initialisation without a governed proxy.

6.3 Django

Scan statistics: 898 files scanned, 2,832 ignored, 3,730 total files across 2,466 directories. Findings: 7 blockers, 0 warnings. All 7 are [FINOS-014, OWASP-002, RBI-018, SEC-007] Shell Injection across management commands, database backends, and translation scripts. An earlier version produced 24 warnings including false positives; after tightening detection patterns to exclude standard framework behaviour, 7 genuine blockers remain. The absence of warnings confirms correct severity calibration.

6.4 OpenSpiel

Scan statistics: 602 files scanned, 1,019 ignored, 1,621 total files across 193 directories. Findings: 4 blockers, 0 warnings. All 4 are [FINOS-014, OWASP-002, RBI-018, SEC-007] Shell Injection for subprocess.Popen calls in game engine integration code. These are architecturally necessary in a game research library; their classification as blockers is correct because in an AI pipeline context where a model influences subprocess command arguments, each is a genuine injection vector. Diamond Cage WASM sandboxing is the appropriate mitigation.

7. Regulatory Compliance Mapping

Table 3 maps Anchor's governance outputs to specific provisions of five regulatory frameworks. A critical property is regulatory polyglottism: a single AuditEntry simultaneously satisfies the logging requirements of all frameworks from a single data structure, without separate implementations per jurisdiction.

Framework	Provision	Anchor Mechanism
EU AI Act	Art 9: risk management	constitution.anchor rule set
EU AI Act	Art 12: logging	DAC AuditEntry chain
EU AI Act	Art 13: transparency	adverse_action_reasons()
RBI FREE-AI	Rec 7: audit trail	DAC + cims_payload()
RBI FREE-AI	Rec 14: explainability	ETH-002 No-Prose Rule + reason codes
RBI FREE-AI	Rec 19: fairness audit	ETH-001 Aho-Corasick taxonomy
CFPB Reg B	Adverse action	adverse_action_reasons()
CFPB Reg B	Reason specificity	ReasonCode field enforcement
NIST AI RMF	Govern 1.1: policies	Signed constitution.anchor
NIST AI RMF	Measure 2.5: provenance	SEC-* violation category
NIST AI RMF	Manage 4.1: monitoring	AnchorRuntime continuous eval
FCA 2024	Oversight records	DAC query API /api/audit
FCA 2024	Model traceability	provider + git_commit in AuditEntry
SEC 2026	AI governance priority	SEC-007 + full domain coverage

Table 3: Regulatory compliance mapping.

8. Discussion

8.1 The Governance Floor Invariant

The invariant that policy.anchor can only raise, never lower, the severity floor of a constitutional rule is central to Anchor's enterprise trust model. An organisation deploying Anchor cannot silently weaken the governance rules it has adopted — cryptographic enforcement makes this structurally impossible rather than merely policy-prohibited. A deploying organisation can demonstrate to a regulator that no developer could have weakened the governance rules their AI system operated under.

8.2 The Governance Invariant Model vs. Probabilistic Classifiers

The ETH domain presents a fundamental challenge: bias detection in natural language output cannot be done with AST queries. Prior approaches use probabilistic classifiers, creating a category error for regulatory contexts — regulators want logical proof, not an 85% confidence score of bias.

Anchor's Governance Invariant model resolves this by shifting from bias detection to structural contract enforcement. ETH-002 enforces the container: unstructured prose decisions are an illegal FLOW by definition. ETH-001 enforces the content: attribution fields are scanned deterministically against a lintable taxonomy, not evaluated probabilistically.

The taxonomy maintenance requirement is an explicit boundary condition, not a weakness. Anchor provides the engine and the grammar; the deploying organisation provides the taxonomy. If the client misses a proxy term, it is a failure of their policy implementation, not the Anchor engine — because Anchor flagged the coverage as partial. This places Anchor in the same category as security scanners like Snyk or Wiz: the platform provides the detection framework, the client provides the ruleset, and the mathematics provides the proof.

8.3 The Notification-as-Evidence Trail

Anchor's audit-not-block architecture is deliberate. Blocking AI decisions in production creates denial-of-service conditions in high-frequency environments. Instead, violations are recorded cryptographically and dispatched to responsible parties via the multi-NOC notification system. The notification receipt — the timestamp at which a compliance officer acknowledged a BLOCKER violation — is itself hashed and appended to the local chain. This creates a human accountability trail alongside the technical audit trail: a regulator can verify not only that a violation occurred, but that the responsible party was notified within seconds.

8.4 Semantic vs. Structural Governance

Anchor's deterministic constitutional layer governs structural and provenance compliance. Subtle semantic violations — hallucinated content that passes structural checks, implicit discriminatory

framing not covered by the taxonomy — require a separate validation tier. Anchor's architecture accommodates this via pluggable semantic validators declared in the manifest:

```
semantic_validators:
  - domain: ETH
    validator: bias_classifier_v1
    tier2_rules: [ETH-001, ETH-002]
    required: false
```

This separation is a feature, not a limitation: it makes the governance boundary explicit, auditable, and honest. The Tier 1 deterministic layer always runs. The Tier 2 semantic layer is opt-in and declared.

8.5 Limitations

- Pattern expressiveness. The current detection layer supports AST queries, regex, and Aho-Corasick trie matching. Probabilistic detectors for subtle semantic violations are not yet supported at Tier 1.
- Model opacity. AnchorRuntime hashes and evaluates model outputs, not model internals. For proprietary black-box models, the runtime cannot verify internal model state. Zero-knowledge proof approaches [11] represent a direction for extending auditability into model internals.
- ETH taxonomy completeness. Prohibited proxy coverage is marked partial for all current concept nodes. Taxonomy expansion is a continuous operational requirement.
- Key management for HMAC signing. The HMAC signing scheme requires secure key distribution to all runtime instances. Integration with hardware security modules (HSMs) or trusted execution environments [10] is recommended for production deployments.
- Lockfile synchronisation on major upgrades. A stale GOVERNANCE.lock triggers an integrity violation. Resolution is anchor init --all --force.

8.6 Future Work: Governance Platform and V5 Primitives

A planned web-based governance platform will provide per-rule audit catalogs with community-contributed mitigation patterns, a telemetry hub for CI/CD report ingestion, and a policy builder GUI for constitution.anchor authoring. The platform implements three visibility scopes (public, organisation, personal) enabling enterprises to contribute private fix patterns without fragmenting the public knowledge base.

V5 introduces a Primitive Vocabulary layer between risk statements and mitigation logic: ACTION, OBJECT, CONTEXT, AUTHORITY, FLOW. Every rule will be decomposable into these typed fields, making the mapping from regulation to detection algorithmic rather than hand-authored. This eliminates the class of errors where a FINOS risk statement is too vaguely worded for the engine to produce complete coverage — the primitive compiler flags incomplete decompositions before they reach the scan engine.

9. Conclusion

We have presented Anchor v4.3.5, a federated governance engine that unifies pre-deployment static code analysis with runtime decision auditability under a single cryptographically signed constitution. The system addresses the dual governance gap facing agentic AI deployments: the absence of constitutional code-level governance and the absence of cryptographically verifiable runtime decision records.

The Diamond Cage extends governance beyond pattern matching to behavioral proof. The GOVERNANCE.lock remote integrity model prevents silent weakening of governance rules. The Decision Audit Chain provides tamper-evident, regulator-queryable records of every AI decision. The Governance Invariant model for the ETH domain replaces probabilistic bias classifiers with deterministic structural contracts, making ethics enforcement as rigorous and auditable as security enforcement. The dual-mode @anchor.enforce decorator makes the governance boundary explicit and API-level: deploying teams declare the guarantees they need, and the system enforces them.

Together, these mechanisms satisfy the logging, explainability, and fairness audit requirements of the EU AI Act, the RBI FREE-AI Report, CFPB Regulation B, the NIST AI RMF, and FCA 2024 guidance from a single data structure. Case studies on four open-source codebases demonstrate that real-world AI governance gaps are pervasive, structurally addressable, and correctly classified by Anchor's federated domain taxonomy.

Anchor v4.3.5 is available as open-source software (Apache 2.0) at github.com/AnimusLab/anchor and on PyPI as pip install anchor-audit.

Acknowledgements

The author thanks the FINOS AI Governance Working Group for early feedback and the proposal by LalaSkye for the primitive vocabulary intermediate layer, and the maintainers of FINOS Architecture-as-Code, HuggingFace Hub, Django, and OpenSpiel for their publicly available codebases.

References

[1] European Parliament and Council. Regulation (EU) 2024/1689 (Artificial Intelligence Act). Official Journal of the European Union, 2024.

[2] Reserve Bank of India. Framework for Responsible and Ethical Enablement of AI (FREE-AI). Technical report, RBI, August 2025.

[3] Consumer Financial Protection Bureau. CFPB orders Apple and Goldman Sachs to pay over $89 million. Enforcement press release, October 2024.

[4] Consumer Financial Protection Bureau. Circular 2024-03: Adverse action notification requirements. CFPB Circular, 2024.

[5] U.S. Securities and Exchange Commission. 2026 Examination Priorities. Technical report, SEC Division of Examinations, January 2026.

[6] National Institute of Standards and Technology. AI Risk Management Framework (AI RMF 1.0). NIST AI 100-1, January 2023.

[7] Financial Conduct Authority. Artificial intelligence: How FCA principles apply. Discussion Paper DP 5/4, FCA, 2024.

[8] Securities and Exchange Board of India. Circular on algorithmic trading governance. SEBI/HO/MRD circular, 2024.

[9] L. Kao. Constant-size cryptographic evidence structures for regulated AI. arXiv:2511.17118, 2025.

[10] Anonymous. Attestable audit: Regulator-verifiable AI audit results. arXiv:2506.23706, 2025.

[11] T. South. Private, verifiable, and auditable AI systems. arXiv:2509.00085, 2025.

[12] I. D. Raji et al. Closing the AI accountability gap. In ACM FAccT, 2020.

[13] J. Mökander et al. Auditing large language models: A three-layered approach. AI and Ethics, 2024.

[14] B. Schneier and J. Kelsey. Secure audit logs to support computer forensics. ACM TISSEC, 2(2):159–176, 1999.

[15] S. Crosby and D. Wallach. Efficient data structures for tamper-evident logging. In USENIX Security, 2009.

[16] NVIDIA. NeMo Guardrails. https://github.com/NVIDIA/NeMo-Guardrails, 2024.

[17] CNCF. Open Policy Agent. https://www.openpolicyagent.org, 2024.

[18] WasmEdge Runtime. https://wasmedge.org, 2024.

[19] M. Brunsfeld et al. Tree-sitter. https://tree-sitter.github.io, 2024.

[20] FINOS Foundation. Architecture as Code. https://github.com/finos/architecture-as-code, 2024.

[21] HuggingFace. huggingface_hub Python Library. https://github.com/huggingface/huggingface_hub, 2024.

[22] Django Software Foundation. Django web framework. https://djangoproject.com, 2025.

[23] M. Lanctot et al. OpenSpiel: A framework for reinforcement learning in games. arXiv:1908.09453, 2019.

[24] B. Chess and G. McGraw. Static analysis for security. IEEE Security & Privacy, 2(6):76–79, 2004.

[25] PyCQA. Bandit: Security linter for Python. https://bandit.readthedocs.io, 2024.

[26] Semgrep. Semgrep: Static analysis at ludicrous speed. https://semgrep.dev, 2024.