Structural, Adversarial, and Availability-Hardened Merkle Architecture for Ternary Moral Logic (TML)

Lev Goukassian
Architect of Ternary Moral Logic
Santa Monica, California

ORCID: 0009-0006-5966-1243

February 14, 2026

Interactive Version: *This document is available as a live web artifact containing Hardened Merkle Architecture formatting.*

1. Merkle as a Core Structural Component of TML

The fundamental architecture of Ternary Moral Logic (TML) departs from the bivalent true/false paradigms of classical computational systems, introducing a mandatory third operational state: the Sacred Zero, or Epistemic Hold.1 This state represents a formal computational hesitation when an artificial intelligence encounters moral ambiguity, relational complexity, or complex ethical thresholds that exceed its confidence parameters.3 However, the philosophical and regulatory imposition of a third state remains meaningless without a verifiable, deterministic enforcement layer. The core proposition of the "Always Memory" pillar dictates an absolute operational axiom: "No Log = No Action".4 To enforce this mandate at the computational level, Merkle tree architecture is not an optional optimization; it is the foundational cryptographic substrate that guarantees the structural integrity of the entire TML framework.5

If Merkle structures are removed from the TML ecosystem, the entire framework's ethical guarantees collapse into mere procedural assertions. Without cryptographic hash commitments and the resulting immutability, a malicious insider with database write access could retroactively alter the logs to conceal a prohibited action (a -1 state) or manually bypass the Epistemic Hold (the 0 state) without leaving a forensic trace.4 The Merkle architecture freezes Sacred Zero outcomes as an immutable moral state. When a system triggers a Sacred Pause, the context, the inputs, and the active ethical parameters are immediately hashed and batched into a Merkle tree, yielding a cryptographic root that serves as an unforgeable evidentiary lock.4

A critical component of this structural integrity is the Active Axiom Set Hash. At the exact nanosecond an event is logged, the exact ruleset, policy weights, and governance constraints governing the AI are hashed and mathematically bound to the event's leaf node.10 This enforces absolute contextual integrity. If a developer attempts to silently modify the schema or alter the moral boundaries post-facto to justify a rogue decision, the Active Axiom Set Hash will immediately mismatch upon independent verification, mathematically proving the discrepancy. Retroactive reinterpretation of an AI's operational state becomes cryptographically impossible.1

Furthermore, the architecture utilizes hierarchical Merkle subtrees to segregate and independently verify distinct ethical domains, specifically Human Rights, Earth Protection, and

Governance.11 This domain-specific branching allows regulators and auditors to inspect ecological compliance without exposing highly sensitive human rights data. The logarithmic nature of Merkle proofs enables highly scalable governance; a massive data center processing millions of autonomous inferences per second can compress its entire moral operational history into a single 32-byte root.9 This proof compression is essential for continuous multi-chain anchoring and limits the bandwidth overhead for third-party verifiers.

Finally, the architecture relies heavily on hash commitments for privacy compliance, specifically enabling crypto-shredding. Personal data is encrypted off-chain, and only the deterministic hash of the ciphertext is anchored in the Merkle tree.9 When General Data Protection Regulation (GDPR) erasure requests are invoked, the specific decryption key is permanently destroyed.12 The raw data is rendered permanently unreadable, fulfilling the erasure mandate, yet the hash remains securely anchored in the Merkle leaf, preserving the continuity and validity of the broader ethical proof chain without leaking sensitive information.13

Consider a concrete adversarial scenario demonstrating this resilience: an autonomous financial agent triggers a flash crash, resulting in massive market instability. A post-mortem review is demanded by regulatory authorities. A malicious infrastructure operator attempts to retroactively re-classify the inference as a hardware fault, modifying the operational database to show a localized memory error rather than a deliberate, algorithmically generated trade. Because the original execution interlock mandated that the inference could only proceed after its Sacred Zero log was committed to a Merkle tree and anchored to a public blockchain 4, the regulator simply requests the Merkle inclusion proof. The modified database generates a new hash that fundamentally fails to align with the anchored Merkle root. The retroactive reinterpretation is definitively blocked, the tamper attempt is forensically flagged, and the original moral context of the decision is preserved for intergenerational review.

2. Canonical Leaf Node Specification (Contextual Integrity Enforcement)

To prevent adversarial manipulation at the serialization layer and ensure cross-platform contextual integrity, every TML moral event must be mapped to a highly rigorous Canonical Leaf Node before hashing. The schema enforces strict immutability, determinism, and privacy guarantees, acting as the atomic unit of the entire governance structure.

Mandatory Fields and Schema Definition

The canonical schema of a TML moral event dictates the inclusion of the following mandatory fields. These fields must be ordered strictly sequentially to ensure deterministic hash outputs across distributed, heterogeneous verification environments.

Field Name	Data Type	Purpose & Security Function
Event ID	UUIDv7	Globally unique, time-ordered identifier designed to prevent replay attacks and ensure monotonic sorting.
Monotonic Sequence ID	uint64	Strict incrementing counter per node to detect truncated or missing log intervals during network-level interception.
Trusted Timestamp	uint64	eIDAS-qualified Unix epoch timestamp (in milliseconds) for rigorous temporal anchoring.14
Sacred Zero Trigger Source	String	The specific neural layer, heuristic, or policy flag that invoked the pause or reflection state.
Pillar Reference	uint8	Enumerated reference to the primary TML pillar involved (e.g., 0x01 for Earth Protection, 0x02 for Human Rights).11
Risk Classification	uint8	Categorical severity mapping generated by the Ethical Uncertainty Score (EUS).15
Reflection Outcome	int8	The final triadic state computed by the state machine: +1 (Permit), 0 (Hold), -1 (Prohibit).1
Integrity Flags	uint16	Bitfield capturing the hardware secure enclave status, multi-signature presence, and memory continuity.

Field Name	Data Type	Purpose & Security Function
Schema Version ID	uint16	Identifies the structural layout of the leaf to prevent parsing errors and synchronization faults.
Schema Hash Commitment	bytes32	Cryptographic hash of the active schema definition itself, binding the data to its structural meaning.
Active Axiom Set Hash	bytes32	Hash of the exact TML rule-set, prompt parameters, and alignment weights active at the precise time of the event.10
Hash Algorithm Version ID	uint8	Identifies the hash function used (e.g., 0x01 for SHA3-512) to enable seamless future post-quantum migrations.16

Active Axiom Set Hash Requirement

The inclusion of the Active Axiom Set Hash provides a continuous, zero-trust defense against silent modifications by internal developers or infrastructure operators. In traditional systems, an AI might make a decision based on an internal policy that is later secretly updated, making the past decision appear anomalous or unjustified under current review. TML treats the rule-set as executable code. Any update to the internal moral axioms, threshold variables, or alignment weights generates a new system-wide hash. Because this specific hash is embedded directly in the leaf node prior to the Merkle commitment, any inference decision is permanently cryptographically bound to the exact rules under which it operated. Regulators can verify whether an AI was operating under an outdated, experimental, or compromised policy by cross-referencing the Active Axiom Set Hash against a public, highly available registry. Retroactive reinterpretation by claiming a different operational policy is mathematically impossible.

Determinism Requirements

To ensure that independent verifiers generate identical hashes from the exact same event data, the architecture mandates strict serialization protocols. A network-level attacker might attempt to

exploit parsing ambiguities to create two structurally different payloads that appear identical to human reviewers but yield different hashes.

- **Canonical Serialization Format:** Payload data must be serialized using strict ASN.1 Distinguished Encoding Rules (DER). This eliminates the optional whitespace and key-ordering ambiguities inherent in standard JSON parsing.
- **Strict Field Ordering:** The fields listed in the canonical schema must be concatenated in the exact byte-order specified before the hash function is applied.
- **Rejection of Non-Deterministic Values:** Floating-point numbers (such as IEEE 754 representations) are strictly prohibited in the leaf node, as different CPU architectures may round these values differently, leading to catastrophic hash divergence. All thresholds, probabilities, and Ethical Uncertainty Scores must be scaled to fixed-point integers (e.g., multiplying a probability by 10^6 and truncating).
- **Locale Independence:** All strings must be strictly UTF-8 encoded without byte-order marks (BOM). Timestamps must be absolute Unix epochs in UTC. Timezone offsets are stripped prior to hashing.

Privacy Requirements

The Canonical Leaf Node is explicitly hardened against the leaking of Personally Identifiable Information (PII) during the hashing and anchoring process. Raw personal data hashing is strictly prohibited due to the vulnerability of MD5, SHA-1, and even SHA-256 to rainbow table inversions and dictionary attacks when the input space (such as names or phone numbers) is highly predictable.

- **Redaction Before Hashing:** Unnecessary PII must be redacted prior to any processing. The redaction process must generate an independent audit trace, proving to regulators that the masking algorithms were applied correctly without exposing the underlying data.
- **Irreversible Pseudonymization:** All required actor identifiers (such as the user prompting the AI) must undergo irreversible pseudonymization via HMAC-SHA3 with a high-entropy, ephemeral hardware-generated salt.
- **Encrypted Payloads:** If detailed payload inclusion is necessary for future forensic reconstruction, the data must be symmetrically encrypted (e.g., AES-256-GCM). Only the ciphertext and the authentication tag are included in the hash preimage.13

Immutability Enforcement

The inclusion of the Schema Hash and Schema Version ID within the leaf guarantees that any alteration to the data structure fundamentally invalidates the cryptographic proof. An external attacker attempting to inject arbitrary data by shifting byte boundaries or exploiting buffer overflows will inherently alter the final leaf hash, instantly breaking the mathematical path to the anchored Merkle root. Any mutation, regardless of its semantic intent, results in total proof failure.

3. Merkle Tree Construction Model (Structural and Security Analysis)

The construction of the TML Merkle tree involves specific cryptographic selections and topological designs meticulously optimized for the triadic nature of the framework and the necessity for extreme high-frequency execution.

Hashing Strategy

The system relies on a dual-hash architecture to balance execution speed, zero-knowledge (ZK) compatibility, and long-term intergenerational security.

- **Explicit Hash Algorithm Selection:** Leaf node contents and payload data utilize SHA3-512 for extreme collision resistance and immunity to length-extension attacks.16 The internal nodes of the Merkle tree utilize SHA-256 to maintain native compatibility with Bitcoin OP_RETURN anchoring scripts and Ethereum virtual machine (EVM) precompiles.16 For specific subsystems requiring highly optimized zero-knowledge proof generation, algebraic hashes such as Rescue or Blake2s are utilized to minimize arithmetic circuit constraints.17
- **Collision Resistance Justification:** Assuming SHA-256 for internal nodes, finding a collision requires an infeasible $O(2^{128})$ operations. A malicious insider with log write access attempting to forge a second-preimage log that hashes to an identical value is bound by computational limits far exceeding current terrestrial capabilities.
- **Migration Path and Post-Quantum Survivability:** Long-term cryptographic degradation is a critical threat. The embedded Hash Algorithm Version ID in the canonical leaf node establishes a forward-compatible migration pathway. As quantum computing capabilities advance toward Shor's algorithm viability, the system can seamlessly transition to Post-Quantum Cryptography (PQC) hash standards, such as SPHINCS+ or XMSS, by incrementing the version identifier. Historical trees generated with SHA-256 remain valid because their roots are perpetually anchored beneath subsequent post-quantum layers, leveraging the security of the newest algorithm to protect the legacy data.

Branching Analysis: Binary vs. Ternary

Traditional blockchain systems predominantly rely on binary Merkle trees. However, TML introduces a ternary state architecture (+1, 0, -1).1 Mathematical modeling of the tree structure reveals significant structural and performance differences when mapping these states to a ternary Merkle tree (a complete tree where every non-leaf node has exactly $d=3$ children).17 In a ternary Merkle tree constructed using a hash function H, an internal node with value v and children v_1, v_2, v_3 takes the computed value $v = H(v_1 \parallel v_2 \parallel v_3)$.17

- **Mathematical Depth Modeling:** The height of a complete binary tree with N leaves is $h_2 = \lceil \log_2(N) \rceil$. The height of a ternary tree is significantly shallower at $h_3 = \lceil \log_3(N) \rceil$.
- **Proof Size Comparison:** To computationally prove inclusion in a binary tree, a verifier requires 1 sibling hash per level. The proof size is $P_2 = h_2 \times \text{hash\_size}$. In a ternary tree, 2 sibling hashes are required per level to reconstruct the parent. The proof size is $P_3 = 2 \times h_3 \times \text{hash\_size}$.19 While the total number of hashes required to construct the tree shrinks to approximately $3/2$ compared to a binary tree, offering a distinct CPU advantage during the massive batch processing required for AI inference 20, the inclusion proof delivered to the light client is marginally larger in total bytes.
- **CPU and Memory Overhead Modeling:** Generating a 10,000-leaf binary tree requires 9,999 hash operations. A ternary tree requires only 4,999 hash operations. In a high-throughput environment running at 10,000 inferences per second, this 50% reduction in internal node hashing directly correlates to lower L2 cache misses and reduced GPU memory bandwidth saturation.

Ternary Geometry Semantic Mapping

Despite the slight increase in network bandwidth for proof transmission, the ternary geometry allows a highly efficient semantic mapping. Subtrees can be structurally clustered by the AI's decision state.
Comparative Structure:

- **Left Branch (v_1):** Aggregates all (+1) Permit inferences (low-risk actions).
- **Center Branch (v_2):** Aggregates all (0) Sacred Pause holds (high ethical complexity).
- **Right Branch (v_3):** Aggregates all (-1) Prohibit decisions (blocked malicious prompts).

This semantic clustering drastically reduces the computational overhead when regulators or independent verifiers specifically query the system. If an auditor requests forensic reconstruction of "all overridden Sacred Zero events" for a specific time window, the traversal algorithm completely ignores the massive left and right branches, isolating its path entirely to the center subtrees. This provides a formal topological justification for the ternary choice.

Construction Requirements

To prevent latency bottlenecks during high-frequency AI inference, the tree construction operates entirely asynchronously:

- **Asynchronous Tree Building and No Inference Blocking:** Events are committed to an in-memory ring buffer. Tree calculation is strictly decoupled from the main neural network

inference thread via direct memory access (DMA) offloading, ensuring zero execution blocking on the critical path.21

- **Rolling Buffer with Integrity Checksum:** The memory buffer is secured by a continuously running checksum. If a memory fault or an insider attack attempts to alter a log sitting in RAM before it is hashed into the tree, the checksum invalidates the buffer and triggers a system halt.
- **Deterministic Leaf Placement:** Leaves are strictly ordered and placed into the tree based on their Monotonic Sequence ID.
- **Odd-Leaf Handling and Balanced Enforcement:** If an anchoring interval terminates with a leaf count that is not a power of 3, the system enforces strict tree balancing by appending mathematically deterministic zero-hashes ($H(0)$) to fill the empty leaf positions.18 This ensures consistent structural depth for all proofs.

Replay Protection

An external attacker or malicious insider might attempt to replay historical favorable decisions or truncate the end of a log sequence to hide malicious activity. This is neutralized by:

- **Event ID Uniqueness Enforcement:** The system rejects any leaf presentation with a previously processed UUIDv7.
- **Strict Monotonic Sequence Validation:** Verifiers confirm that $Seq_{n} = Seq_{n-1} + 1$.
- **Replay Detection Logic:** A bloom filter maintains a rolling window of the last 10 million Event IDs. Any collision immediately halts processing and flags the input as a highly probable replay attack.

4. Hierarchical Integrity Model

To optimize forensic reviews, scale efficiently, and strictly compartmentalize sensitive data based on regulatory purview, the TML Merkle implementation relies on a sophisticated Hierarchical Integrity Model. Rather than flatly hashing all diverse events into a single chaotic layer, the system maintains independent subtrees dedicated to specific ethical domains defined by the TML framework.11

Domain Isolation and Master Root Aggregation

The architecture mandates the separation of logs into distinct subtrees:

- **Human Rights Subtree:** Isolates decisions pertaining to bias, discrimination, user privacy, and censorship. Access to this subtree typically requires high-level cryptographic clearance due to potential PII remnants and legal sensitivities.
- **Earth Protection Subtree:** Aggregates decisions concerning environmental impact, carbon burn rates, data center cooling optimizations, and resource allocation

constraints.11 This subtree is highly transparent and generally open to public audit via standard APIs.
- **Governance Subtree:** Tracks infrastructure-level changes to the axioms, schema updates, access control modifications, and hardware enclave health attestations.

These independent subtrees are periodically aggregated into a single, higher-order "Root-of-Roots," or Master Root. The Master Root computation includes a Root Versioning ID to track structural schema changes at the highest architectural level without breaking legacy verification protocols.

Forward Integrity Safeguards

A persistent threat in any logging system is a malicious infrastructure operator with full administrative access attempting a long-range attack: retroactively fork-mining an entire parallel history of the AI's operations to cover up a systemic failure that occurred weeks prior. To defeat this, the Master Root incorporates continuous forward integrity safeguards.

- **Strictly Increasing Root Index:** Every Master Root contains a strictly increasing interval integer.
- **Prior Root Hash Inclusion:** The computation of the current Master Root must mathematically encompass the hash of the immediately preceding Master Root.

Let R_{n} be the calculated root for the current interval n, representing the combined state of the Human Rights, Earth Protection, and Governance trees. The anchored bound root B_{n} is mathematically defined as:
$B_{n} = H(B_{n-1} \parallel R_{n} \parallel \text{Timestamp})$
This establishes an unbreakable forward hash chain. An attacker wishing to alter a specific event at interval $n-5$ must not only break the collision resistance of the Merkle tree for that specific batch but must also sequentially recalculate and successfully re-anchor B_{n-4}, B_{n-3}, B_{n-2}, B_{n-1}, and B_{n}. Because these bound roots are pushed to distributed blockchains, this creates absolute computational irreducibility. The past is mathematically sealed.

5. Anchoring Strategy (Time-Bound Enforcement)

The "Hybrid Shield" represents the external defensive layer of TML, designed to protect the integrity of the moral history against localized 51% attacks, nation-state censorship, or catastrophic central database corruption.4 It achieves this through a highly aggressive multi-chain blockchain anchoring protocol.1

Time-Bound Enforcement

An infrastructure operator could theoretically attempt to suppress an immoral AI action by indefinitely delaying the anchoring of the specific Merkle root that contains the incriminating leaf. To defeat this "silent delay" attack vector, the architecture defines a strict, numerical maximum anchoring delay of **2000 milliseconds**.

- **Automatic Anchoring Trigger Mechanism:** The asynchronous in-memory buffer automatically flushes and computes a Master Root every 2 seconds, or when the batch size exceeds 10,000 events, whichever condition is met first.
- **Multi-Chain Anchoring Strategy:** Relying on a single ledger introduces platform-specific risk. The resulting 32-byte hash is simultaneously broadcast via highly available RPC nodes to:
    - **Polygon (Layer 2):** For high-speed, low-cost verification, confirming presence within seconds and acting as the real-time watchdog.9
    - **Ethereum (Layer 1):** For medium-term robust security, leveraging massive decentralized consensus, and enabling the execution of automated penalty smart contracts via the Memorial Fund.1
    - **Bitcoin (via OpenTimestamps):** For ultimate, nation-state-level immutability and intergenerational preservation.1
- **Monotonic Root Indexing:** Each anchored payload broadcast to these chains includes the strict index counter and the forward-linked hash.

Time Integrity Requirements

To prevent anti-backdating, the system enforces Trusted Timestamp integration. The eIDAS-qualified timestamp within the Root-of-Roots is cryptographically bound to the payload. During verification, this internal timestamp is strictly cross-referenced against the block header timestamp of the underlying blockchain (e.g., the Ethereum block time).14 If the internal timestamp claims the event occurred on Tuesday, but the blockchain anchor timestamp registers on Thursday, a temporal anomaly is instantly flagged.

Reconciliation Protocol

Independent light nodes and governance watchdogs operate public verification endpoints. These nodes continuously poll the Polygon network for the latest TML anchors.

- **Detection of Missing Root Intervals:** If node N observes an anchor with Sequence ID 1000 and the next observed anchor on the chain is Sequence ID 1002, a missing root interval is mathematically detected.
- **Mandatory Anomaly Logging:** This triggers an automated, high-priority anomaly log sent to the overarching Governance Council.

- **Independent Audit Pathway:** The infrastructure operator is immediately placed on probation and must initiate an independent audit pathway. They must either produce the missing Merkle root for Sequence ID 1001 and successfully prove its contents, or face automatic cryptographic slashing of their staked assets via smart contract execution.10

6. Causal Integrity Enforcement (Sacred Zero Protection)

The fundamental ethical promise of the TML standard is the "Sacred Zero"—a moment of forced hesitation and reflection when an AI encounters profound ethical ambiguity.3 A devastating, highly sophisticated attack vector involves a malicious insider or a compromised model silently bypassing this pause state, rendering an unethical decision, and then forging a plausible log post-facto.

Causal Ordering Proof Requirement

To absolutely prevent this bypass, the architecture mandates an Execution Interlock at the hardware level.4 The system must structurally demonstrate that the commitment of the log to the Merkle tree occurs *before* or *atomically with* the release of the final inference output to the downstream application or user.

- **Atomic Snapshot Boundary:** The boundary is defined at the exact sub-millisecond clock cycle where the neural network's activation thresholds dictate the transition to a 0 or +1 state.7 If a +1 (Permit) is achieved, the tensor output is frozen and held in an isolated, secure hardware buffer.
- **Formal Ordering Guarantee:** The causal link is enforced by generating a deterministic hash dependency between the output packet and the logging system.

Execution Interlock Mechanism

1. The AI inference engine formulates the decision and generates the trace data.
2. The trace data is routed via an internal bus to the isolated Always Memory daemon.
3. The daemon constructs the Canonical Leaf Node, hashes it using SHA3-512, and injects it into the active Merkle buffer.
4. Crucially, the daemon returns a cryptographically signed receipt containing the exact leaf hash to the inference engine.
5. **Output Encapsulation:** The inference output payload is encapsulated for network transmission. The encapsulation software explicitly requires the signed leaf hash to compile the final network packet.
6. **Prevention of Silent Bypass:** If the egress router or API gateway attempts to process an output packet that lacks the corresponding, HSM-signed log hash, the packet is instantly dropped as structurally invalid.10

Through this deterministic hardware dependency, the log mathematically predates the action. "No Log = No Action" transitions from a highly aspirational administrative policy to a physical, code-level constraint that cannot be circumvented without entirely rewriting the network stack.4

7. Proof Generation and Verification

The decentralized governance power of TML relies heavily on lightweight, widely distributed verification. Regulators, civil rights organizations, or third-party auditors must be able to verify specific historical AI events without downloading the petabytes of operational data generated by a major platform's daily inference load.4

Light Client / SPV Specification

The verification model utilizes a Simplified Payment Verification (SPV) approach uniquely adapted for the TML ternary architecture. A regulator investigating a specific autonomous decision requests a targeted proof bundle from the operator.
Independent Third-Party Verification Workflow:

1. **Inputs Required:** The verifier receives a small JSON package containing the raw Event Payload, the standardized Merkle Inclusion Proof (an array of sibling hashes leading to the root), and the Anchored Root Reference (the specific transaction ID on the Polygon or Ethereum blockchain).
2. **Local Hashing:** The verifier's standard laptop or mobile device locally hashes the Event Payload according to the strict Canonical Leaf Node serialization rules.
3. **Path Traversal:** The verifier iteratively hashes the resulting leaf against the provided sibling hashes to reconstruct the candidate root. Because the tree depth is strictly logarithmic ($O(\log_3 N)$), even validating an event from a massive batch of 1 billion logs requires a proof size of only a few kilobytes, easily processed in milliseconds.19
4. **Blockchain Anchor Confirmation:** The verifier queries an independent, public blockchain RPC node to confirm that the locally reconstructed root exactly matches the root permanently anchored in the specified block transaction.

Verification Failure Handling: Verification mathematically fails if the hashes mismatch, the sequence ID is anomalous, the eIDAS timestamp is outside the block window, or the blockchain anchor cannot be found. Any failure is treated as absolute proof of structural tampering or operator non-compliance.
Furthermore, the architecture leverages Sparse Merkle Trees (SMTs) with an astronomical 2^{256} leaves to generate definitive *non-inclusion proofs*.17 If an operator claims a specific prohibited payload was never processed by their systems, an SMT proof can mathematically demonstrate the absence of that exact payload hash in the historical tree, shifting the burden of proof away from ambiguous log analysis to undeniable cryptography.17

Key Security and Compromise Detection

The cryptographic strength of the leaf hashes is deeply reliant on the security of the underlying keys used for pseudonymization and encryption.

- **Ephemeral Encryption Keys:** Keys are generated dynamically per batch or per session.
- **Hardware-Backed Storage:** All keys are generated and stored exclusively within FIPS 140-2 Level 3 compliant Hardware Security Modules (HSMs). Keys never touch standard disk storage or system RAM in plaintext.
- **Key Rotation Schedule:** The system mandates a strict, automated key rotation schedule every 24 hours, or immediately following the detection of an anomalous access pattern.
- **Compromise Detection Protocol:** The HSMs are monitored by external threshold signature nodes. If an HSM detects unauthorized physical access or memory probing, it instantly zeroizes its internal key material and broadcasts a signed "Integrity Frozen" state to the governance blockchain, halting all dependent AI inferences globally.10

Crypto-Shredding Mechanism

Public blockchains are permanently transparent, but AI logs often contain highly sensitive personal data. This creates a severe structural conflict with privacy regulations like the GDPR's "Right to Erasure." TML resolves this paradox elegantly with crypto-shredding.9

1. Before any personal data enters the Merkle pipeline, it is symmetrically encrypted using AES-256-GCM.
2. The HSM generates a highly ephemeral decryption key specifically for that dataset.
3. The *ciphertext* is hashed, and this hash is what populates the Canonical Leaf Node.13
4. When a valid GDPR data erasure request is received, the operator initiates the destruction of the specific ephemeral decryption key held in the HSM.
5. Once the key is destroyed, the ciphertext becomes mathematically impossible to decrypt. The underlying personal data is completely, irreversibly obliterated.
6. **Preservation of Hash Continuity:** Crucially, because the deterministic hash of the ciphertext remains structurally unaltered in the Merkle leaf, the integrity of the root, the batch, and the overall historical evidence chain is perfectly preserved.13 Continued proof validity post-erasure is guaranteed, ensuring the AI cannot use privacy regulations as a convenient loophole to delete evidence of its broader moral reasoning patterns.12

8. Data Availability (DA) Strategy

A valid Merkle root mathematically proves that specific data existed at a certain precise point in time, but it does not store or transmit the data itself.5 If an external attacker compromises the

operator's storage servers, or a catastrophic natural disaster causes total data loss, the regulator might possess a perfectly valid Merkle root on the Ethereum blockchain but be completely unable to read the underlying encrypted pre-hash event data necessary for forensic reconstruction.

Explicit Mandate: A Merkle root without retrievable, readable data completely fails TML governance guarantees. The system defaults to an assumption of "spoliation of evidence" if the pre-image logs cannot be produced upon regulatory request.14

Storage, Redundancy, and Distribution Model

To defend against catastrophic storage compromise or systemic hardware failure, the architecture enforces a highly redundant, geographically distributed Data Availability (DA) strategy.

Storage Architecture	Redundancy Mechanism	Access Latency	Primary Threat Mitigation
Centralized Cloud (Tier 1)	Multi-AZ replication (e.g., AWS S3)	< 10 ms	High-frequency read queries; mitigated localized node failure.
Decentralized Network (Tier 2)	IPFS / Arweave with erasure coding	500 ms - 2s	Defeats state-level censorship and centralized cloud provider outages.
Optical WORM Storage (Tier 3)	Physical geographic distribution in deep cold storage	Hours / Days	Defeats ransomware, insider deletion, and electromagnetic pulse (EMP) events.

- **Retention Horizon Specification:** Moral Trace Logs mapping to low-risk (+1) permit states are retained for a rolling window of 7 years. Logs mapping to Sacred Zero (0) holds or high-risk prohibitions (-1) are subject to intergenerational retention mandates, stored on Write-Once-Read-Many (WORM) optical archives for a minimum of 50 years.
- **Proof-of-Storage Attestation:** The decentralized storage networks must periodically generate zero-knowledge Proofs of Spacetime (PoSt). This mathematical attestation guarantees to the master TML smart contract that the distributed nodes are actively holding the data, mitigating the severe risk of silent storage degradation or node abandonment over time.
- **Disaster Recovery and Data Rehydration Workflow:** In a severe disaster recovery scenario where Tier 1 storage is wiped, the infrastructure operator initiates a rehydration workflow. The system pulls the erasure-coded fragments (e.g., Reed-Solomon chunks)

from the Tier 2 decentralized network, mathematically reassembles the ciphertext, decrypts it using the surviving HSM-backed keys, and hashes it to prove perfect structural alignment with the blockchain-anchored Merkle root. Once verified, the data is pushed back to the operational Tier 1 storage.

9. Log Truncation and Tamper Resistance

An advanced insider attempting to rewrite operational history might not try the mathematically impossible task of forging a single leaf. Instead, they might attempt to quietly truncate the last 5,000 logs of an active batch *before* they are anchored to the blockchain, attempting to hide a catastrophic AI safety failure that occurred at the very end of a session.

Append-Only Storage Enforcement

TML defends against truncation via strictly append-only database semantics enforced at the core kernel level of the storage servers. The storage drives are configured to physically reject any overwrite or delete commands outside of the authorized crypto-shredding key destruction workflow.

- **Periodic Integrity Checks:** Background daemons continuously run verification tasks, validating the $Seq_n = Seq_{n-1} + 1$ rule across the entire active storage cluster.
- **Automatic Anomaly Signaling:** A missing sequence ID or a detected truncation attempt immediately halts the AI's inference engine and fires an automated, high-priority anomaly signal to the independent TML Governance Council.10 The system enters a hard "Integrity Frozen" state that requires manual, multi-signature cryptographic override to resume.10

Schema Governance

The structural layout of the data itself represents a subtle attack vector. The Schema Hash embedded in every leaf prevents spontaneous restructuring 24, but organized schema updates are occasionally necessary as models evolve.

- **Signed Schema Registry:** All valid schema definitions are hosted in a public, cryptographically signed registry.
- **Dual Control for Updates:** Updating the schema requires a multi-signature cryptographic approval (e.g., 7 of 13 keys from an independent oversight board).10 A single rogue developer cannot alter the logging structure.
- **Independent Anchoring:** The new schema definition is independently hashed and anchored to the blockchain. When parsing a leaf, the SPV software pulls the schema definition corresponding to the Schema Version ID, hashes it, and strictly compares it to the Schema Hash Commitment in the leaf. If they differ by even a single byte, the verification fails entirely.

10. Latency and Throughput Modeling

A critical and highly publicized objection to the TML framework is the "Alignment Tax"—the argument that mandatory cryptographic hashing, state buffering, logging, and Merkle root evaluation will cripple the speed of modern high-frequency AI models.23 The architecture formally mandates the preservation of a ≤ 2 ms user-visible inference latency while sustaining an extreme event rate of $\ge 10,000$ events per second.14

Computational Overhead and Worst-Case Load Model

At a throughput of 10,000 decisions/second, sequentially hashing 10,000 complex payloads on a single CPU thread would vastly exceed the 2 ms budget. This latency bottleneck is solved via a highly optimized, parallel, dual-lane hardware architecture.25

1. **Lane A (Inference Execution):** The AI evaluates the inputs and computes the final output tensor.
2. **Lane B (Governance Execution):** Simultaneously, the Sacred Pause controller and logging buffer process the ethical state via parallel DMA channels.

Because the massive Merkle tree construction is completely asynchronous, the system only needs to block execution long enough to calculate the initial *Canonical Leaf Node hash* to satisfy the Execution Interlock requirement.
Numerical Scenario Example ($\ge 10,000$ events/sec):

- Generating a SHA3-512 hash for a standard 4 KB output payload takes approximately 0.05 ms on modern NVIDIA Tensor Core GPU architectures.25
- Digital signature generation (Ed25519) by the HSM takes 0.1 ms.
- Memory copy overhead across the PCIe bus takes 0.05 ms.
- Total computational overhead added directly to the critical path is strictly bounded below 0.5 ms.25
- Assuming the core AI inference takes 1.2 ms, the total processing time is 1.7 ms, safely meeting the ≤ 2 ms visible latency cap.

Parallel Construction Strategy

Behind the scenes, the asynchronous Merkle construction algorithm processes massive volumes without impacting the user. By grouping events into a tree, $10,000$ leaves can be condensed into a single root via exactly $4,999$ internal SHA-256 hashing operations (for a ternary topology).17 Utilizing parallel SIMD (Single Instruction, Multiple Data) instructions on the GPU, generating the entire 10,000-leaf tree takes less than 1.5 ms of background processing time.25 The final root is then seamlessly forwarded to the network broadcast pool for multi-chain

anchoring. This rigorously demonstrates that the framework scales under worst-case load models without violating operational speed constraints.

11. Formal Integrity Guarantees

The immense architectural resilience of TML relies on defining precise, formal cryptographic assumptions and understanding exactly when those assumptions degrade.

- **Collision Resistance Assumptions:** It is computationally infeasible to find two different canonical leaf nodes that produce the exact same hash. Under SHA-256 and SHA3-512, this guarantee holds far beyond any current terrestrial computational limits, protecting against malicious log substitution.
- **Preimage and Second-Preimage Resistance:** An external attacker cannot reverse-engineer a Sacred Zero log simply by looking at the blockchain anchor, nor can a malicious insider find a substitute log that hashes to an existing valid leaf already in the tree.
- **Forward Integrity Definition:** Mathematically, an event recorded and anchored at time T cannot be invalidated, altered, or obfuscated by any action taken at time $T+1$. The strictly unidirectional flow of the hash chain ensures the past is sealed.

Conditions of Guarantee Degradation

These formal guarantees degrade under two highly specific, catastrophic conditions:

1. **Quantum Supremacy Arrival:** Shor's Algorithm running on a sufficiently large, fault-tolerant quantum computer could break the underlying elliptic curve digital signatures (like ECDSA or Ed25519) used in the execution interlock and Ethereum smart contracts, allowing for forged hardware attestations.
2. **Catastrophic Hash Function Compromise:** The discovery of a rapid, mathematically devastating collision attack against SHA-256, rendering it as broken as MD5.

To model and ensure long-term survivability, the system embeds version identifiers explicitly to allow a Post-Quantum migration. As the National Institute of Standards and Technology (NIST) finalizes PQC standards, the TML Master Root will dynamically transition to anchoring via hash-based signatures (e.g., XMSS or SPHINCS+), which are structurally immune to quantum algorithms, ensuring the integrity of the moral history intergenerationally.

12. Comparative Analysis

To properly contextualize the TML Merkle architecture, its specific design tradeoffs must be rigorously compared against existing, well-known cryptographic accumulator frameworks operating in adjacent domains.

Feature Matrix	TML Ternary Merkle Architecture	Bitcoin Transaction Merkle Trees	Ethereum State Trie (Patricia)	Certificate Transparency (CT) Logs	Sparse Merkle Trees (SMT)
Primary Goal	Verifiable ethical accountability 26	Financial double-spend prevention	Global state machine synchronization	TLS certificate issuance audit	Efficient non-inclusion proofs
Branching Topology	Ternary ($d=3$) 18	Binary ($d=2$) 17	Hexadecaphonic ($d=16$)	Binary ($d=2$)	Binary (2^{256} leaves) 17
Leaf Content	Triadic Moral Event schema 1	Unspent Transaction Outputs (UTXOs)	Account balances / Smart contract storage	X.509 Certificates	Key-Value pairs
Scalability Limit	Highly scalable via asynchronous parallel processing 25	Block size constraint (1-4 MB)	Storage bloat due to state transition complexity	Massive append-only scalability	Computationally heavy but high sparse optimization
Audit Clarity	Exceptional (semantic clustering of 0 states)	Low (requires full block parsing for context)	Moderate (path proofs are complex)	High (strictly chronologically ordered)	High (direct key querying)
Governance Robustness	Unprecedented (Execution interlock, Hybrid Shield) 1	High (Decentralized consensus)	High (Consensus + Slashing)	Moderate (Relies on external monitors)	Dependent on underlying host

While Ethereum provides excellent state synchronization, its hexadecaphonic (16-branch) structure creates unnecessary computational overhead for purely append-only event logging. CT logs provide excellent append-only properties but completely lack the critical execution interlock required for real-time AI safety. The TML approach explicitly trades a slight increase in individual proof size (due to the ternary structure) 19 for a massive, systemic gain in audit clarity and profound semantic alignment with the (+1, 0, -1) moral states.

13. Failure Mode Disclosure

Despite extensive adversarial hardening, rigorous mathematical modeling, and highly redundant multi-chain anchoring, the TML Merkle architecture carries explicit residual risks that must be fully disclosed to regulatory and governance institutions.

- **Explicit Residual Risk Statement:** The framework's absolute immutability guarantees fail entirely if all three layers of the multi-chain Hybrid Shield (Polygon, Ethereum, Bitcoin) suffer simultaneous, coordinated 51% attacks, allowing deep blockchain reorganizations.9 While the economic and logistical cost of such a coordinated, global attack is prohibitively astronomical, it remains mathematically possible.
- **Cryptographic Dependency Transparency:** A more realistic catastrophic failure mode involves a day-zero vulnerability discovered in the hardware security modules (HSMs) managing the ephemeral pseudonymization keys. If an advanced persistent threat (APT) or nation-state actor successfully exfiltrates the HSM entropy pool, they could theoretically decrypt the crypto-shredded PII before the erasure mandates are executed, resulting in a massive privacy breach.
- **Data Loss Impact Assessment:** Finally, total data loss across both the primary Tier 1 centralized storage and the Tier 2 decentralized erasure-coded networks results in a critical scenario where the cryptographic proofs survive, but the context perishes. While the AI operator is still severely penalized by the spoliation assumption 14, the forensic capability to actually diagnose *why* a catastrophic AI failure occurred is permanently lost. The architectural mandate, therefore, remains relentlessly focused on aggressive replication, hardware interlocking, and intergenerational optical storage to ensure that the moral history of autonomous intelligence remains a permanent, auditable record for rigorous human oversight.

Document Control

Attribute	**Detail**
Document ID	TML-SEC-001
Version	1.0.0
Status	**AUDIT READY**
Classification	**Security Hardening**
Author	Lev Goukassian

Attribute	Detail
Date	2026-02-14

Works cited

1. FractonicMind/TernaryMoralLogic: I've always believed that the hardest problems in AI aren't technical; they're architectural. We keep building systems that can't explain themselves, can't prove their own integrity, can't handle uncertainty without either freezing or lying. And then we act surprised when - GitHub, accessed February 14, 2026, https://github.com/FractonicMind/TernaryMoralLogic
2. Ternary Logic (TL): Evidentiary Framework for Economic Systems - SSRN, accessed February 14, 2026, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5663410
3. TernaryMoralLogic/theory/philosophical-foundations.md at main - GitHub, accessed February 14, 2026, https://github.com/FractonicMind/TernaryMoralLogic/blob/main/theory/philosophical-foundations.md
4. Technical Architecture & Governance of TML Smart Contracts: A Deterministic Enforcement Layer for Ternary Moral Logic : r/solidity - Reddit, accessed February 14, 2026, https://www.reddit.com/r/solidity/comments/1qjil7f/technical_architecture_governance_of_tml_smart/
5. Technical Architecture & Governance of TML Smart Contracts - SSRN, accessed February 14, 2026, https://papers.ssrn.com/sol3/Delivery.cfm/5985954.pdf?abstractid=5985954&mirid=1
6. TernaryMoralLogic/docs/General_FAQ.md at main · FractonicMind, accessed February 14, 2026, https://github.com/FractonicMind/TernaryMoralLogic/blob/main/docs/General_FAQ.md
7. Adversarial Audit & IP Fortress for Ternary Moral Logic (TML) | by Lev Goukassian - Medium, accessed February 14, 2026, https://medium.com/@leogouk/adversarial-audit-ip-fortress-for-ternary-moral-logic-tml-b4dac4139f2e
8. TernaryMoralLogic/TML_Pillars/Sacred_Zero ... - GitHub, accessed February 14, 2026, https://github.com/FractonicMind/TernaryMoralLogic/blob/main/TML_Pillars/Sacred_Zero-Sacred_Pause_Technology.md
9. TernaryMoralLogic/TML_Pillars/Public_Blockchains.md at main - GitHub, accessed February 14, 2026, https://github.com/FractonicMind/TernaryMoralLogic/blob/main/TML_Pillars/Public_Blockchains.md
10. Technical Architecture & Governance of TML Smart Contracts: The Deterministic Enforcement Layer for Ethical AI - SSRN, accessed February 14, 2026, https://papers.ssrn.com/sol3/Delivery.cfm/5985974.pdf?abstractid=5985974&mirid=1

11. Ternary Moral Logic (TML) - Ethical AI Framework, accessed February 14, 2026, https://fractonicmind.github.io/TernaryMoralLogic/
12. Verifiable AI Refusal Events using SCITT Transparency Logs - IETF Datatracker, accessed February 14, 2026, https://datatracker.ietf.org/doc/draft-kamimura-scitt-refusal-events/
13. Key transparency and the right to be forgotten | Hacker News, accessed February 14, 2026, https://news.ycombinator.com/item?id=42210605
14. Ternary Moral Logic (TML) Quantitative Governance Analysis | by Lev Goukassian - Medium, accessed February 14, 2026, https://medium.com/@leogouk/ternary-moral-logic-tml-quantitative-governance-analysis-d874812eb158
15. From Principles to Verifiable Implementation: A Policy-Technical Analysis on Operationalizing the UNESCO Recommendation on the Ethics of Artificial Intelligence with Ternary Moral Logic, accessed February 14, 2026, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5649910
16. TernaryMoralLogic/docs/MANDATORY.md at main · FractonicMind, accessed February 14, 2026, https://github.com/FractonicMind/TernaryMoralLogic/blob/main/docs/MANDATORY.md
17. blockchain-foundations.pdf, accessed February 14, 2026, https://www.marabu.dev/blockchain-foundations.pdf
18. VERI-ZEXE: Decentralized Private Computation with Universal Setup - ResearchGate, accessed February 14, 2026, https://www.researchgate.net/publication/361681892_VERI-ZEXE_Decentralized_Private_Computation_with_Universal_Setup
19. (PDF) Merkle Hash Grids Instead of Merkle Trees - ResearchGate, accessed February 14, 2026, https://www.researchgate.net/publication/344603589_Merkle_Hash_Grids_Instead_of_Merkle_Trees
20. Merkle Hash Grids Instead of Merkle Trees - e-Publications@Marquette, accessed February 14, 2026, https://epublications.marquette.edu/cgi/viewcontent.cgi?article=1043&context=comp_fac
21. Auditable AI: Tracing the Ethical History of a Model, accessed February 14, 2026, https://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID5655110_code8713860.pdf?abstractid=5655110&mirid=1&type=2
22. Hot Chips - gibsonnet.net, accessed February 14, 2026, http://gibsonnet.net/aix/ibm/mmi202102.issue.pdf
23. The Oracle of the Sacred Zero. Pause when truth is uncertain. Refuse… | by Lev Goukassian | Feb, 2026 | Medium, accessed February 14, 2026, https://medium.com/@leogouk/the-oracle-of-the-sacred-zero-083b014a03d7
24. Ternary Moral Logic (TML) and the EU AI Act: A Technical Backbone for Enforceable Accountability - SSRN, accessed February 14, 2026, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=5655090

25. Ternary Moral Logic (TML) and the Future of AI Governance: A Technical Analysis for NVIDIA - SSRN, accessed February 14, 2026, https://papers.ssrn.com/sol3/Delivery.cfm/5856362.pdf?abstractid=5856362&mirid=1
26. accessed February 14, 2026, https://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID5655090_code8713860.pdf?abstractid=5655090&mirid=1#:~:text=We%20present%20Ternary%20Moral%20Logic,an%20AI's%20decision%2Dmaking%20process.
27. Merkle Hash Grids Instead of Merkle Trees - Thomas Schwarz SJ, accessed February 14, 2026, https://tschwarz.mscs.mu.edu/Papers/Mascots20.pdf