Cryptographic Audit Receipts for Trustworthy AI

A Metadata-Centric Framework Using Lazy Capsule Materialization (LCM) and Merkle Anchors

Denzil James Greenwood
founder@cognitiveinsight.ai

Version 1.0 – September 14, 2025

Table of Contents

Contents

Version 1.0 – September 14, 2025

Cryptographic Audit Receipts for Trustworthy AI

A Metadata-Centric Framework Using Lazy Capsule Materialization (LCM) and Merkle Anchors

---

This paper is authored by Denzil James Greenwood as part of ongoing research and development of CognitiveInsight.ai. It is intended to inform discussions on verifiable AI governance and compliance frameworks.

Denzil James Greenwood

founder@cognitiveinsight.ai

Abstract

Artificial Intelligence (AI) has become foundational in domains where compliance, accountability, and trust are non-negotiable. Yet traditional cryptographic mechanisms—focused on keys for encryption and authentication—do not by themselves solve the problem of verifiable lineage and auditability across complex AI pipelines. This paper introduces the Cognitive Insight Audit Framework (CIAF) and its core innovation, Lazy Capsule Materialization (LCM). CIAF LCM uses cryptographically bound metadata, Merkle trees, and selective proof generation to provide sub-second, tamper-evident audit receipts for any artifact in the AI lifecycle: dataset records, model checkpoints, or inferences. Unlike systems that log exhaustively by default, LCM commits to metadata leaves and anchors them via signed Merkle roots. Proof capsules can then be materialized on demand, ensuring both verifiability and efficiency while preserving privacy and intellectual property.

---

1. Introduction

The governance of AI is constrained by an "auditability gap." Regulators demand transparency; auditors require robust records; and developers must protect both sensitive data and proprietary models. Traditional logging systems produce descriptive, mutable records, while cryptographic security mechanisms primarily safeguard confidentiality (via encryption) and identity (via signatures). Neither directly addresses the need for verifiable provenance of AI training and inference processes.

CIAF LCM closes this gap by transforming compliance evidence into cryptographically verifiable commitments. Instead of retaining full logs or datasets, CIAF commits to

canonicalized metadata for every relevant artifact, organizes those commitments into Merkle trees, and anchors the resulting root under signed policy statements. Verification is selective: a proof capsule can be generated on demand to confirm the inclusion of any data point, training epoch, or inference in the tamper-evident record. This design aligns with regulatory requirements while preserving scalability, privacy, and performance.

---

2. Background: Keys in Classical Cryptography

• Symmetric keys: a shared secret enables both encryption and decryption (e.g., AES).

• Asymmetric keys: paired public/private keys enable encryption, signatures, and verification (e.g., RSA, ECC).

• Session keys: ephemeral symmetric keys negotiated via asymmetric exchange for efficient communication.

These key systems excel at protecting data confidentiality, ensuring integrity, and authenticating actors. However, they do not directly answer the question: "Can we prove what dataset, model, and inference were used in this AI system, at a specific point in time, under a given policy?"

---

3. CIAF LCM: Anchors Instead of Keys

- **Keys** secure secrets and authenticate signers.
- **Anchors** commit to canonicalized metadata for datasets, models, and inferences, binding them into a verifiable chain of evidence.

Whereas key-centric systems prove *who sent what*, CIAF LCM proves *what was used, produced, and committed, under which policy, and when*.

Lazy Capsule Materialization (LCM). Unlike systems that store full proofs by default, CIAF LCM materializes proof capsules only when required. A proof capsule contains the metadata, its Merkle path, and the signed anchor. This approach minimizes storage overhead while still ensuring that verifiable, tamper-evident receipts can be generated on demand.

---

4. Metadata as the Fundamental Unit

4a. Metadata Capture

All CIAF audit data is captured in metadata leaves. Each leaf represents one artifact (data record, model snapshot, or inference) encoded in canonical JSON, ensuring deterministic hashing.

Examples:

Dataset Record Metadata

```
{
 "dataset_family": "credit_approval",
 "split": "train",
 "record_id": "row_12345",
 "features_hash": "sha256:af34...",
 "timestamp": "2025-09-14T10:22Z",
 "policy": "CIAF_v1.1"
}
```

Model Epoch Metadata

```
{
 "model_id": "xgboost_credit_v1.0",
 "epoch": 5,
 "checkpoint_hash": "sha256:b932...",
 "loss": 0.072,
 "policy": "CIAF_v1.1"
}
```

Inference Metadata

```
{
 "model_id": "xgboost_credit_v1.0",
 "input_hash": "sha256:a1b2...",
 "output_hash": "sha256:c3d4...",
 "inference_id": "2025-09-14T10:25Z#001",
 "policy": "CIAF_v1.1"
}
```

Each metadata blob is hashed, producing a leaf node in the Merkle tree.

Note. Metadata can be defined to include extensive information about the blob. Bias, drift, data curation, and process tags can be added at any point in the pipeline to help evaluate the model against regulatory or business concerns. While this adds some computational overhead, the value of these checks generally outweighs the cost.

An example metadata specification, including optional fields for datasets, models, inferences, and anchors, is provided in Appendix A.

4b. Metadata Storage

CIAF LCM ensures that all metadata commitments are stored in a way that balances performance, durability, and verifiability. The storage layer is designed with three objectives:

• Canonicalized Integrity: Each metadata blob is first canonicalized (deterministic JSON representation) and hashed before storage. This guarantees that even if multiple systems or teams generate metadata, the stored representation is uniform and hash-consistent.

• Dual Anchoring (Hash Table + Merkle tree): Metadata is first written into a hash table for immediate indexing (O(1) lookup) and then rolled up into the Merkle tree structure for global integrity. The hash table acts as the fast-access layer, while the Merkle tree acts as the cryptographic ledger layer.

• Write-Once, Read-Many (WORM) discipline: Stored metadata is never overwritten. Instead, new commitments are appended with a timestamp, preserving the lineage of changes. This provides immutability guarantees consistent with regulatory expectations for audit logs.

• Storage backends: CIAF LCM is storage-agnostic: metadata commitments can be persisted in relational databases, object storage (e.g., S3, GCS), blockchain layers, or secure transparency logs. The only requirement is that the storage backend maintains append-only semantics and supports replay for verification.

• Metadata selection policy: CIAF distinguishes between required fields (baseline compliance) and optional fields (extended assurance). The actual stored metadata therefore reflects both the minimum necessary to prove lineage and the additional fields organizations configure to meet domain-specific regulatory needs.

In short: Metadata storage in CIAF LCM is not just about retention — it is about anchoring verifiable commitments in multiple layers (hash table, Merkle tree, WORM log, optional external anchor) so that every artifact in the AI lifecycle can be retrieved, verified, and proven against regulatory and business requirements.

4c. Required vs. Optional Metadata

Not all metadata fields need to be committed in every scenario. CIAF LCM distinguishes between required fields, which form the minimum auditability baseline, and optional fields, which organizations can include to satisfy sector-specific regulations, internal governance, or customer trust requirements.

Required Fields (Baseline Auditability):
These are essential for every commitment and provide the minimal cryptographic evidence needed to prove provenance, accountability, and policy alignment.

- Unique identifier (record ID, model ID, inference ID)
- Policy reference (e.g., CIAF_v1.1)
- Timestamp of commitment
- Cryptographic hash of the features, checkpoint, or input/output pair

- Actor identity (*who* initiated the action; e.g., user ID, service account, or signing key)
- Execution context (*where* the action occurred; e.g., data center, jurisdiction, or cloud region)
- System/machine identifier (*what machine* or environment performed the operation; e.g., host fingerprint, container ID, or TEE attestation)

Optional Fields (Extended Assurance):
These fields provide richer compliance context and can be selectively included based on regulatory obligations or business goals.

- Bias and fairness metrics (e.g., demographic parity, disparate impact)
- Drift detection statistics (e.g., KL divergence between training and live data)
- PII checksums (flagging presence of sensitive attributes at ingestion)
- Training performance metrics (loss, accuracy, F1 at specific epochs)
- Business rule validations (e.g., credit approval thresholds, safety rules in healthcare)
- Hardware/software configuration details beyond the required system ID (e.g., driver versions, GPU model)
- Cryptographic attestations from MPC/TEE/ZKP systems (optional but can be anchored as digests)

Example Scenarios:

- A financial regulator may require at minimum the actor, timestamp, and system context for each model decision, ensuring accountability.
- A healthcare auditor may additionally require fairness metrics, PII validation, and device identifiers for FDA audit trails.
- An internal AI governance team may configure drift detection and fairness metrics to be logged automatically at each batch ingestion step.

Design Principle:
This separation ensures that CIAF LCM delivers efficiency (only the minimum is always logged) and adaptability (extra context can be logged when required). By doing so, CIAF enables organizations to right-size their metadata commitments while preserving the *who, what, where, when, and under which policy* — the core elements regulators demand for accountability.

---

5. Merkle Tree Construction

To aggregate commitments, CIAF LCM constructs a Merkle tree:

• Leaf hashing: each metadata commitment is hashed into a leaf.

Example: `h1 = H(leaf1)`, `h2 = H(leaf2)`, …, `hn = H(leafn)`

• Pairwise parent hashing: leaf hashes are paired and concatenated, then re-hashed.

Example:

`parent = H(h_left || h_right)` (if there is an odd number of leaves at a level, the last hash is duplicated to maintain balance).

• Iterate until root: the process repeats recursively until a single hash — the Merkle root — remains.

Example: `root = H(parentAB || parentCD || …)`

The final Merkle root is a compact cryptographic fingerprint of all leaves in the tree. Any change to any metadata leaf alters the root, ensuring tamper-evidence across the entire structure.

---

6. Anchoring and Signing

The Merkle root is then bound into an anchor:

Example:

Anchor = (root || policy_id || schema_version || timestamp || domain_labels)

Signature = Sign(Anchor, private_key)

• The anchor, along with its digital signature, is appended to a write-once log (WORM). This ensures that once a commitment is recorded, it cannot be modified without detection.

• (Optionally, multiple CIAF roots can be batched into a higher-level "Merkle-of-Merkles" root, which may then be anchored externally to a public timestamping service or blockchain ledger for added transparency.)

---

7. From Root to Proof: Inclusion Verification

To verify a single artifact:

• Retrieve its metadata blob.

• Recompute its leaf hash.

• Provide the Merkle path (sibling hashes along the path to the root).

• Reconstruct the Merkle root.

• Compare against the signed anchor root.

• Verify the anchor's signature and timestamp.

If all checks pass, the artifact is cryptographically proven to belong to the committed set, under the specified policy and at the recorded time.

---

8. Lazy Capsule Materialization (LCM)

CIAF differs from systems that pre-store full proofs. Instead:

• Default: only anchors and append-only logs are retained.

• On demand: when a regulator or auditor requests verification, CIAF materializes a proof capsule, consisting of the artifact's metadata, its Merkle path, and the signed anchor.

• Result: storage savings by orders of magnitude, sub-second proof generation, and minimal disclosure (only the relevant artifact and path are exposed, not the entire dataset or model).

---

9. Comparison with Traditional and Emerging Approaches

Note. The following subsections highlight selected examples of recent work in the field of AI verifiability. They are included only to provide points of comparison with the CIAF LCM process. The intent is not competition, but rather to situate CIAF LCM within the broader ecosystem of approaches and to illustrate its complementary role.

9a. Key-Centric Cryptography vs. CIAF Anchors

Aspect	Key-Centric Cryptography	CIAF LCM Anchors
Goal	Confidentiality, authentication	Provenance, verifiability
Unit	Keys (symmetric/asymmetric)	Metadata leaves + anchors
Output	Ciphertexts, signatures	Audit receipts, proof capsules

Verification	Who sent what	What data/model/inference existed when
Disclosure	Full message/signature	Selective proof on demand
Weakness	Storage/logging external	Anchors don't encrypt content (complementary)

Why Anchors. CIAF LCM adopts anchors because they extend cryptographic assurances from *actors* (who signed what) to *artifacts* (what data, model, or inference was committed, under which policy, and when). Keys remain essential for securing identities and signatures, but anchors provide tamper-evident lineage across the AI lifecycle. In short, anchors complement keys by shifting the cryptographic guarantee from *secrecy and authentication* toward *provenance and auditability*, which is the missing layer in current AI governance frameworks.

Anchors can incorporate required accountability fields (actor_id, system_id, location/jurisdiction), aligning cryptographic provenance with regulatory expectations around 'who, what, where, when, and under which policy.'

9b. Relation to Zero-Knowledge–Based Verifiable Pipelines

Recent work has explored the use of zero-knowledge proofs (ZKPs) to achieve verifiability across the AI pipeline. For example, Balan, Learney, and Wood (2025) propose a framework for end-to-end verifiable AI pipelines, leveraging proofs of training (ZKPoT), inference (ZKPoI), and unlearning (ZKPoU). Their goal is to prove that each stage of the pipeline was executed correctly, while maintaining data and model privacy.

CIAF LCM addresses a complementary challenge. Instead of proving computational correctness of every operation, CIAF focuses on canonicalized metadata commitments, Merkle anchoring, and Lazy Capsule Materialization. This ensures regulators and auditors can obtain tamper-evident receipts of what data, model, or inference was used, under which policy, and at what time, with sub-second proof generation.

These approaches are not mutually exclusive. CIAF LCM can be extended to incorporate ZK proofs as optional metadata fields, anchoring their digests within the CIAF Merkle structure. In this way, organizations can start with lightweight, scalable provenance tracking, while adding stronger ZK-based assurances in high-risk contexts such as regulated benchmarks or model fairness audits.

9c. Relation to MPC and TEE-Based Approaches

Beyond zero-knowledge proofs, other cryptographic and hardware-assisted approaches have been proposed for AI auditability, including multi-party computation (MPC) and Trusted Execution Environments (TEEs). These systems aim to ensure that model training and inference occur in secure, verifiable environments, often with guarantees of confidentiality and composability. For example, recent work (Balan, Learney, and Wood, 2025) explores end-to-end frameworks that combine zkSNARKs, MPC, and TEEs to produce proofs of training, inference, and unlearning.

While these approaches provide strong assurances of correctness and confidentiality, they remain computationally intensive, frequently requiring specialized hardware or incurring high latency. CIAF LCM takes a complementary path: instead of proving every computation cryptographically, it anchors canonicalized metadata in Merkle trees and materializes proof capsules on demand. This yields lightweight, scalable auditability with sub-second verification — a design aligned with regulatory and operational needs where efficiency and practicality are paramount.

Crucially, CIAF LCM is designed to integrate with advanced cryptography. MPC- or TEE-generated attestations can be logged as optional metadata fields, with their digests anchored into the CIAF Merkle structure. This allows organizations to balance operational feasibility with formal cryptographic strength, adopting hybrid models in high-assurance domains such as defense, finance, or healthcare.

9d. Complementary Nature of Approaches

CIAF LCM does not aim to replace advanced cryptographic or hardware-assisted methods. Instead, it provides a lightweight, scalable baseline for auditability that can stand alone or integrate with more computationally intensive techniques such as ZKPs, MPC, or TEEs. This layered approach reflects real-world deployment realities: most organizations benefit from efficient provenance tracking in daily operations, while reserving stronger proofs for high-risk or regulated scenarios. Where ZK/MPC/TEE attestations exist, CIAF records their digests as optional fields, enabling cross-system verification without duplicating heavy proofs.

---

10. Regulatory Recap and CIAF Mapping

The following summarizes how CIAF LCM directly addresses major regulatory requirements in AI governance and related sectors. Each regulation defines specific concerns around risk, transparency, oversight, and accountability; CIAF maps these concerns into cryptographically verifiable commitments.

EU AI Act (2024)

Regulatory Concern:

• Ensure data quality and governance (Article 10).

• Maintain traceability through record-keeping (Article 12).

• Document and enforce risk management (Article 9).

• Provide human oversight (Article 14).

CIAF Alignment:

• Commits datasets, splits, and model metadata as canonicalized leaves → proves data provenance.

• Anchored Merkle roots = tamper-evident, permanent records → satisfies record-keeping.

• Metadata extensions (bias metrics, PII checks, drift detection) support continuous risk management.

• Human-in-the-loop checkpoints can be embedded at commitment points → verifiable oversight.

NIST AI Risk Management Framework (2023)

Regulatory Concern:

• Provide accountability for AI governance.

• Ensure transparency of inputs, assumptions, and data context.

• Enable measurement and monitoring of bias, drift, and accuracy.

• Ensure risk controls and mitigations are documented and repeatable.

CIAF Alignment:

• Anchors create cryptographic receipts for all model lifecycle events → governance proof.

• Metadata captures dataset family, splits, features → ensures transparency.

• Optional commitments (bias, drift, fairness metrics) → measurable evidence of monitoring.

• Logs + anchors capture mitigations and policy adjustments → permanent audit trail of risk controls.

ISO/IEC 42001 (AI Management System Standard)

Regulatory Concern:

• Document operational planning and control of AI systems (Clause 8).

• Provide performance evaluation (Clause 9).

• Demonstrate continuous improvement (Clause 10).

CIAF Alignment:

• Each dataset, model, and inference is committed as metadata, proving operational control.

• Performance metrics (loss, accuracy, F1) can be committed as optional fields → verifiable evaluation.

• Retraining and corrective actions are cryptographically anchored → evidence of ongoing improvement.

Healthcare Regulations (HIPAA, FDA, EMA)

Regulatory Concern:

• Protect patient privacy while maintaining data traceability.

• Prove that AI-assisted diagnostics meet safety/quality standards.

CIAF Alignment:

• Anchors commit to hashed representations of sensitive data → proof without exposure.

• Metadata commitments can log quality/safety checks → auditable without violating privacy.

• Optional PII checksums prove compliance with de-identification rules.

Financial Regulations (SEC, FINRA, MiFID II)

Regulatory Concern:

• Ensure fairness and transparency in trading and credit decision models.

• Provide immutable audit trails for regulators.

CIAF Alignment:

• Anchored audit receipts for each inference/decision → immutable proof of compliance.

• Bias/fairness metrics committed as metadata → regulators can verify fairness testing occurred.

• Logs and capsules allow selective disclosure of transactions to regulators.

Defense / Government (FedRAMP, DoD AI Guidelines)

Regulatory Concern:

• Ensure AI models run in approved, controlled environments.

• Provide tamper-evident logs of training, deployment, and inference events.

CIAF Alignment:

• Anchors bind models, checkpoints, and deployments to signed policies → proof of approved configuration.

• Append-only logs + Merkle roots guarantee tamper-evidence for mission-critical models.

• Anchored policies can include algorithm/hardware environment specifications.

Summary Table

Regulation	Concern	CIAF LCM Response
EU AI Act	Data quality, traceability, risk management, oversight	Dataset commitments, anchored logs, bias/PII checks, human-in-loop checkpoints
NIST AI RMF	Governance, transparency, measurement, risk management	Anchors for lifecycle events, metadata context, optional bias/drift metrics
ISO/IEC 42001	Planning, evaluation, improvement	Metadata for control, performance metrics, retraining anchors
Healthcare (HIPAA/FDA/EMA)	Privacy + safety	Hashed data commitments, PII checks, quality logs
Finance (SEC/FINRA/MiFID II)	Fairness, audit trails	Anchored receipts for inferences, bias/fairness metadata, selective disclosure
Defense/Government	Approved configs, tamper-evidence	Anchored policies, append-only logs, environment metadata

11. Applications

CIAF LCM is broadly applicable across domains where auditability, trust, and efficiency are critical. Key applications include:

- **AI Governance and Regulatory Compliance**
  Regulators can request proof capsules that verify compliance with legal and ethical requirements without requiring access to raw datasets or proprietary

models. This provides regulators with confidence while protecting trade secrets.

- **Healthcare and Finance**
  In high-stakes sectors where privacy, fairness, and safety are paramount, CIAF enables auditors to validate decision-making processes using tamper-evident proofs while ensuring sensitive data and personal identifiers remain protected.

- **Model Intellectual Property Protection**
  Model builders can demonstrate regulatory compliance and accountability by exposing cryptographic receipts instead of revealing raw model weights, parameters, or training data, thereby safeguarding competitive advantage.

- **Scalable Audit Logging**
  Traditional audit trails generate terabytes of logs that are costly to store and analyze. CIAF replaces exhaustive logs with lightweight, on-demand proof capsules, enabling sub-second compliance checks at scale without storage bloat.

- **Cross-Organizational Assurance**
  CIAF facilitates trust between organizations (e.g., vendors, regulators, auditors, and customers) by providing cryptographically bound, independently verifiable evidence of AI behavior, creating a shared language of proof across stakeholders.

---

12. Security Properties

CIAF LCM assumes a strong adversary capable of tampering, partial disclosure, or replay attacks. Its layered design defends against these threats through append-only anchoring, inclusion proofs, and optional independent timestamping. Key properties include:

- **Tamper-Evidence**
  Any modification to a metadata leaf alters its hash, which propagates upward and changes the Merkle root. Because the root is cryptographically anchored and signed, even the smallest manipulation becomes immediately detectable.

- **Selective Disclosure**
  Proof capsules reveal only the requested metadata, its Merkle path, and the signed anchor. This minimizes unnecessary exposure of sensitive or proprietary information while still enabling independent verification.

- **Algorithm Agility**
  Anchors are not bound to a single cryptographic primitive. If a hash or signature algorithm is deprecated, commitments can be re-anchored under newer algorithms without losing historical verifiability.

- **Independent Time Proofs**
  Anchors can be optionally cross-linked to external timestamping services, transparency logs, or blockchain ledgers, providing resilience against backdating or replay attacks and enhancing third-party audit credibility.

- **Resilience to Operational Failures**
  Because commitments are written under append-only (WORM) discipline and supported by redundant hash-table + Merkle tree anchoring, CIAF can detect, localize, and recover from system-level inconsistencies without compromising integrity.

Result: CIAF LCM ensures that verifiability is not only theoretical but practically robust, delivering operational trustworthiness across regulatory, auditing, and production environments.

12a. Collision Resistance and Risk Mitigation

CIAF LCM relies on collision-resistant hash functions such as SHA-256 for its metadata commitments and Merkle tree construction. While the probability of two distinct metadata blobs producing the same hash is vanishingly small (on the order of 1 in 2^128 for birthday-bound collisions), the framework implements layered safeguards to further reduce operational risk:

- **Dual Anchoring**

  Every metadata blob is written into both a hash table and a Merkle tree. A collision would have to persist consistently across both structures to escape detection.

- **Append-Only Discipline**

  Commitments are stored under write-once, read-many (WORM) rules. If a collision were ever detected, the affected root would immediately invalidate, preventing silent tampering.

- **Algorithm Agility**

  CIAF anchors can be migrated to newer hash functions (e.g., SHA-3, BLAKE3) as cryptographic practice evolves, ensuring long-term resilience.

- **Independent Verification**

Proof capsules include the metadata, its Merkle path, and the signed anchor, allowing verifiers to recompute commitments independently and detect inconsistencies.

These properties make collision-based attacks not only computationally infeasible but also practically unexploitable, reinforcing CIAF LCM's role as a robust, trustworthy audit framework.

---

13. Conclusion

CIAF LCM extends classical cryptographic ideas beyond secrecy and identity into the domain of verifiable AI auditability. By anchoring metadata leaves into Merkle trees, binding them with signed roots, and materializing proofs only when needed, it provides regulators, auditors, and AI builders with a shared, scalable, privacy-preserving framework.

Keys remain essential: they secure the actors by protecting identities and authenticating signers. Anchors, however, secure the evidence — they cryptographically bind datasets, models, and inferences into tamper-evident commitments under explicit policies and timestamps. Crucially, metadata can be configured to capture not only *what* was committed, but also *who* initiated the action, *where* it occurred, and on *which system or environment* it was executed. This ensures that regulatory requirements for accountability ("who, what, where, when, under which policy") can all be met in a cryptographically verifiable form.

Together, keys and anchors deliver the dual foundation for trustworthy AI: keys establish who acted; anchors prove what was committed, where and when, under which policy.

Disclaimer on Regulatory Alignment

CIAF LCM provides a cryptographic framework that can support compliance with regulatory requirements by enabling verifiable commitments to datasets, models, policies, and oversight processes. However, the degree to which any specific regulatory obligation is satisfied depends on the scope and quality of metadata implemented within the framework.

In practice:

- **Baseline compliance** is possible through required metadata fields (identifiers, timestamps, policies, hashes).

- **Extended compliance** requires organizations to include relevant optional metadata fields (e.g., bias metrics, drift detection, PII validation, business rules) as needed for their sector or jurisdiction.
- **Human oversight remains essential**: CIAF can cryptographically prove that oversight points were logged, but it is ultimately up to human reviewers, auditors, and regulators to interpret results, make decisions, and enforce corrective actions.

Therefore, CIAF should be understood as a **compliance-enabling tool, not a compliance guarantee**. Responsibility for regulatory alignment remains with the implementing organization, which must configure CIAF appropriately and ensure meaningful human oversight.

The CIAF LCM process is and will be improved on a continuous basis to meet the needs of the community.

Abbreviations

• CIAF – Cognitive Insight Audit Framework

• LCM – Lazy Capsule Materialization

• TEE – Trusted Execution Environment

• WORM – **Write-Once, Read-Many (WORM)**

• RMF – Risk Management Framework

References

• Merkle, R. (1988). A Digital Signature Based on a Conventional Encryption Function. In: Advances in Cryptology — CRYPTO '87. Lecture Notes in Computer Science, vol 293. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-48184-2_32

• National Institute of Standards and Technology (NIST). (2023). Artificial Intelligence Risk Management Framework (AI RMF 1.0). U.S. Department of Commerce. https://doi.org/10.6028/NIST.AI.100-1

• European Commission. (2024). Artificial Intelligence Act. Regulation (EU) 2024/1689 of the European Parliament and of the Council. Official Journal of the European Union. https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng

• International Organization for Standardization (ISO). (2023). ISO/IEC 42001: Artificial Intelligence Management System. Geneva: ISO/IEC.

• U.S. Department of Health & Human Services (HHS). (2013). Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule. 45 CFR Part 160 and Subparts A and E of Part 164.

• U.S. Securities and Exchange Commission (SEC). (2023). Regulation Best Interest: The Broker-Dealer Standard of Conduct. 17 CFR §240.15l-1.

• Financial Industry Regulatory Authority (FINRA). (2023). Regulatory Notice 23-12: Artificial Intelligence in Financial Services. Washington, DC.

• U.S. Department of Defense (DoD). (2020). Ethical Principles for Artificial Intelligence. Defense Innovation Board.

- South, T. (2025). *Private, Verifiable, and Auditable AI Systems*. PhD Thesis, MIT. arXiv+1

- Balan, Kar; Learney, Robert; Wood, Tim. (2025). *A Framework for Cryptographic Verifiability of End-to-End AI Pipelines*. arXiv. arXiv+1

- Souza, R., Azevedo, L., Mattoso, M. et al. (2019). *Provenance Data in the Machine Learning Lifecycle in Computational Science and Engineering*. IEEE/ACM WORKS. ResearchGate+1

Appendix A: Metadata Specification for CIAF LCM

This appendix provides a structured list of required and optional metadata fields for CIAF LCM commitments. The distinction enables baseline verifiability while allowing organizations to tailor metadata commitments to regulatory, sector-specific, or internal governance requirements.

A.1 Dataset Record Metadata

Required

- **dataset_family**
- **split**
- **record_id**
- **features_hash**

- **timestamp**
- **policy**
- **actor_id (*who created or ingested the record*)**
- **system_id (*what machine or service processed the record*)**
- **location/jurisdiction (*where the action took place*)**

Optional

- **label_hash**
- **bias_metrics**
- **pii_check**
- **source_system**
- **curation_process**
- **regulatory_domain**
- **hardware_env details**

A.2 Model Epoch / Checkpoint Metadata

Required

- **model_id**
- **epoch**
- **checkpoint_hash**
- **timestamp**
- **policy**
- **actor_id (*who ran the training step*)**
- **system_id (*what machine/environment executed the training*)**
- **location/jurisdiction**

Optional

- **training_loss**

- **accuracy / f1_score**
- **drift_stats**
- **explainability_summary**
- **regularization/params**
- **energy_consumption**
- **extended hardware/software config (e.g., GPU type, driver version)**

A.3 Inference Metadata

Required

- **model_id**
- **inference_id**
- **input_hash**
- **output_hash**
- **timestamp**
- **policy**
- **actor_id (*who or what process requested the inference*)**
- **system_id (*what machine produced the result*)**
- **location/jurisdiction**

Optional

- **confidence_score**
- **fairness_context**
- **pii_check**
- **explanation_hash**
- **business_rule_applied**
- **decision_override**
- **audit_reference**

A.4 Anchor Metadata (Root-Level)

Required

- **root**
- **policy_id**
- **schema_version**
- **timestamp**
- **domain_labels**
- **signature**
- **signing_key_id (*who signed the anchor*)**

Optional

- **batch_root**
- **external_anchor (e.g., blockchain, transparency log)**
- **auditor_id**
- **retention_policy**
- **jurisdiction (e.g., GDPR, HIPAA compliance zone)**
- **environment attestations (TEE/MPC/ZKP proofs)**

Appendix B: Example Proof Capsule (JSON)

This example illustrates a Lazy Capsule Materialization (LCM) proof generated on demand. It demonstrates how metadata, Merkle path, and anchors are combined into a verifiable unit.


```
{
 "capsule_type": "inference_proof",
 "metadata": {
  "model_id": "xgboost_credit_v1.0",
  "inference_id": "2025-09-14T10:25Z#001",
```

```
  "input_hash": "sha256:a1b2...",
  "output_hash": "sha256:c3d4...",
  "policy": "CIAF_v1.1",
  "timestamp": "2025-09-14T10:25Z"
 },
 "merkle_path": [
  "sha256:deadbeef...",
  "sha256:22334455...",
  "sha256:998877aa..."
 ],
 "anchor": {
  "root": "sha256:112233...",
  "policy_id": "CIAF_v1.1",
  "schema_version": "1.0",
  "timestamp": "2025-09-14T10:30Z",
  "domain_labels": ["CIAF|model", "CIAF|inference"],
  "signature": "ed25519:abcdef..."
 }
}
```


This JSON capsule can be independently verified by recomputing the leaf hash, reconstructing the Merkle root via the supplied path, and validating the anchor signature.

Appendix C: Verifier Checklist

The following checklist outlines the steps and inputs required to verify a proof capsule.

Inputs Required

- **Metadata blob (canonical JSON)**
- **Merkle path (sibling hashes to the root)**
- **Anchor (root, policy, schema_version, timestamp, domain_labels, signature)**
- **Public key of signer**

Verification Steps

1. **Canonicalize the metadata into JSON and compute its hash.**
2. **Rebuild the Merkle root using the leaf hash and provided path.**
3. **Compare roots: reconstructed root must equal the root in the anchor.**
4. **Validate signature: check that the anchor's signature verifies against the signer's public key.**
5. **Check timestamp and policy: ensure the anchor timestamp is valid and the policy_id matches the expected CIAF schema.**
6. **Verify accountability fields (required for regulatory contexts):**
    - **actor_id: confirm *who* performed the action is recorded.**
    - **system_id: confirm *what machine or environment* executed the action.**
    - **location/jurisdiction: confirm *where* the action occurred.**
7. **Optional checks: verify sector-specific metadata such as fairness metrics, PII validation, performance thresholds, or extended system attestations (TEE, MPC, ZKP digests).**

Result
If all checks pass, the capsule is cryptographically proven to be part of the committed set, under the stated policy, and at the recorded time — with accountability for *who, where, and what machine* included in the verification.

Appendix D: Reference Implementation (GitHub Repository)

The Cognitive Insight Audit Framework (CIAF) and Lazy Capsule Materialization (LCM) are actively maintained as open research and development projects. A reference implementation and supporting examples are published on GitHub:

Repository: github.com/DenzilGreenwood/pyciaf

License: MIT

Contents:

• /ciaf – Core CIAF LCM Python modules (metadata capture, Merkle tree, anchoring, proof generation)

• /examples – Demonstrations including credit model demo and quickstart guide

• /docs – Technical documentation, architecture notes, and compliance mapping

• /tests – Unit and integration tests for reproducibility and verifiability

• /tools – Utility scripts for receipt verification and framework validation

How to Use:

```
# Clone the repository

git clone https://github.com/DenzilGreenwood/pyciaf.git

cd pyciaf

# Install the framework (modern Python packaging)

pip install -e .

# Run the credit approval model demo

python examples/credit_model_demo.py

# Run the quickstart example

python examples/quickstart.py

# Inspect generated audit receipts under examples/outputs/

ls examples/outputs/

# Verify receipts independently

python tools/verify_receipt.py examples/outputs/credit_receipt_*.json
```

The repository uses modern Python packaging with pyproject.toml and will evolve with contributions, including performance benchmarks, JSON schema definitions for proof capsules, and integration guides for enterprise AI governance pipelines.

Founder, CognitiveInsight.ai

Date: September 14, 2025