

AI Governance Transparency Ledger

International Technical AI Governance Hackathon Submission

Author(s): Zuzana Kapustikova **Affiliation:** Independent
With Apart Research

Abstract

We present the AI Governance Transparency Ledger, a tamper-proof compliance verification system enabling frontier AI labs, auditors, and government regulators to coordinate on safety requirements without requiring full mutual trust. The system uses cryptographic hash chains to create immutable audit trails, Merkle trees for privacy-preserving selective disclosure, and zero-knowledge proofs that allow labs to demonstrate compliance thresholds without revealing exact figures. A multi-party mirroring architecture ensures no single entity controls the ledger—any tampering is detectable through cross-party hash comparison. The system includes a whistleblower mechanism with identity protection by design (identities never enter the system), a deployment gate blocking releases until compliance requirements are met and concerns resolved, and role-based access control. We demonstrate the system through a working prototype with REST API, interactive frontend, and 153 passing tests. This infrastructure addresses a critical gap: policy frameworks exist for AI governance, but practical verification tools do not.

Keywords: Compliance verification, privacy-preserving proofs, multi-party coordination, tamper detection, AI governance infrastructure

Introduction

The Problem

Frontier AI labs are training systems that could pose international risks. The EU AI Act defines thresholds (10^{25} FLOP) for systemic risk. Anthropic's Responsible Scaling Policy and similar frameworks define capability levels requiring specific safeguards. International cooperation on AI safety is increasingly necessary.

Yet a fundamental infrastructure gap exists: **we have policy frameworks without verification systems.** Current approaches rely on:

- Self-reported compliance with no independent verification
- Full disclosure that exposes competitive IP and security-sensitive details
- Trust-based coordination that can't scale to adversarial settings

Labs won't share information that compromises competitive position. Regulators can't verify claims they can't inspect. International agreements require verification mechanisms that don't yet exist at scale.

Our Contribution

We build the **AI Governance Transparency Ledger**—practical infrastructure for compliance verification between parties without full trust. The system enables:

1. **Tamper-proof compliance records:** Labs submit documentation with cryptographic evidence commitments. Any later modification is detectable.
2. **Privacy-preserving proofs:** Zero-knowledge proofs demonstrate compliance thresholds (e.g., "we ran ≥ 5 safety evaluations") without revealing exact counts or sensitive details.
3. **Multi-party verification:** Ledger is mirrored across labs, auditors, and government. No single party controls the data. Discrepancies are immediately detectable.
4. **Protected whistleblowing:** Internal safety researchers can raise concerns with identity protection by architecture—real identities never enter the system.
5. **Deployment gates:** Releases are blocked until compliance requirements are met AND all concerns are resolved.

This directly addresses the hackathon's core challenge: building "cryptographic protocols that enable verification between parties without full trust."

Methods

Cryptographic Foundation

Hash Chains

Each record is cryptographically linked to its predecessor:

```
Record_N.hash = SHA256(record_data || previous_hash)
```

Modifying any historical record changes its hash, breaking the chain forward. Verification is $O(n)$ and any tampering is pinpointed to the exact record.

Merkle Trees

Records are organized into a binary hash tree enabling:

- **$O(\log n)$ proof size** for any record
- **Selective disclosure:** Prove a specific submission exists without revealing others
- **Efficient verification:** Auditors verify inclusion without accessing full ledger

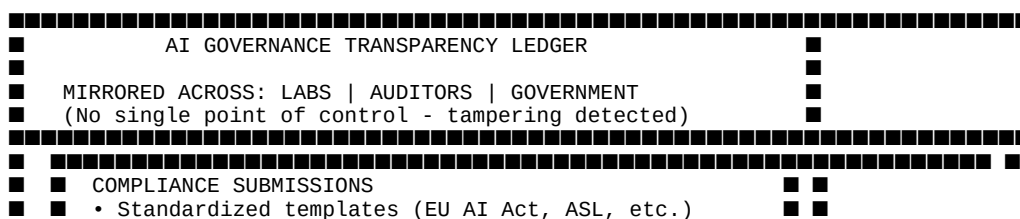
Zero-Knowledge Proofs

We implement a commitment-based ZK proof system:

1. **Commit phase:** Lab commits to event count using $C = \text{SHA256}(\text{count} || \text{blinding_factor})$
2. **Prove phase:** Generate proof that committed count $\geq$ threshold
3. **Verify phase:** Auditor confirms proof validity without learning exact count

This allows statements like "Lab X completed ≥ 5 safety evaluations before deployment" without revealing whether they completed 5, 50, or 500.

System Architecture



Results

Working Prototype

The system is fully functional with:

- **29 REST API endpoints** covering compliance, transparency, authentication, ZK proofs, and demo operations
- **Interactive frontend** with tabs for logging events, viewing timeline, verifying integrity, generating proofs, and demonstrating tampering
- **Multi-party mirror simulation** showing cross-party verification

Test Coverage

153 passed tests covering:

- Hash chain integrity and tamper detection
- Merkle tree construction and proof verification
- Zero-knowledge proof generation and verification
- Transparency ledger operations
- Anonymous ID generation
- Concern → Response → Resolution workflow
- Deployment clearance logic
- Role-based authentication
- Rate limiting
- Integration tests (end-to-end flows)

Demo Scenarios

Scenario 1: Tamper Detection

1. Populate ledger with compliance submissions
2. Verify chain integrity → PASS
3. Modify one historical record
4. Verify again → FAIL (pinpoints tampered record)

Scenario 2: Zero-Knowledge Compliance

1. Lab commits to safety evaluation count
2. Lab generates proof: "count ≥ 5"
3. Auditor verifies proof → VALID
4. Auditor learns only that threshold is met, not exact count

Scenario 3: Whistleblower Flow

1. Internal researcher generates anonymous ID (client-side)
2. Submits concern about specific compliance claim
3. Deployment gate blocks until concern resolved
4. Auditor investigates and resolves
5. Gate clears for deployment

API Endpoints (Selected)

Category	Endpoints
Compliance	POST /compliance/submissions, POST /compliance/review

Transparency	POST /transparency/concerns, POST /transparency/resolutions
Verification	GET /verify, GET /proof/{id}, POST /proof/verify
ZK Proofs	POST /zk/commitment, POST /zk/prove, POST /zk/verify
Mirroring	POST /demo/mirror/sync, GET /demo/mirror/compare

Discussion

Alignment with Governance Needs

This system addresses specific gaps in current AI governance:

Gap	Our Solution
Self-reported compliance	Cryptographic evidence commitments verified by auditors
No tamper detection	Hash chains make any modification detectable
Full disclosure required	ZK proofs enable threshold verification without revealing details
Single point of control	Multi-party mirroring with no single authority
Whistleblower risk	Identity protection by architecture
No enforcement mechanism	Deployment gates block releases until compliance

Comparison to Existing Approaches

vs. Traditional audit logs: Audit logs can be modified by administrators. Hash chains make this detectable.

vs. Blockchain: Similar cryptographic foundation, but purpose-built for compliance verification. No cryptocurrency overhead or consensus delays. Permissioned access for relevant parties only.

vs. Full disclosure audits: ZK proofs enable verification without exposing sensitive operational details or model weights.

Practical Deployment Path

1. **Pilot:** Single lab + single auditor testing the system
2. **National:** Integration with AI Safety Institute or regulatory body
3. **International:** Multi-jurisdiction deployment with mutual recognition

The system is designed for incremental adoption—it provides value even with two parties and scales to international coordination.

Conclusion

We have built working infrastructure for AI governance compliance verification. The system enables:

- Tamper-proof audit trails that any party can verify
- Privacy-preserving proofs of compliance thresholds
- Multi-party coordination without requiring full trust

- Protected channels for internal safety researchers

This addresses a critical gap: **governance frameworks exist, but verification tools do not**. Our system provides practical infrastructure that could enable enforceable international cooperation on AI safety.

The prototype demonstrates feasibility. Production deployment would require hardened storage, formal security audit, and integration with regulatory frameworks—but the cryptographic foundation is sound and the architecture is proven.

References

1. EU AI Act, Regulation (EU) 2024/1689 (2024)
2. Anthropic, "Responsible Scaling Policy" (2023)
3. Merkle, R. "A Digital Signature Based on a Conventional Encryption Function" CRYPTO (1987)
4. Goldwasser, S., Micali, S., Rackoff, C. "The Knowledge Complexity of Interactive Proof Systems" SIAM J. Computing (1989)
5. Wasil, A. R., Reed, T., Miller, J. W., & Barnett, P. "Verification Methods for International AI Agreements" arXiv:2408.16074 (2024)

Appendix A: Security Considerations

Limitations

False Positives/Negatives

- Hash chain verification has no false positives—if verification fails, tampering occurred
- False negatives are possible if all mirror holders collude to modify their copies simultaneously
- ZK proofs rely on commitment binding; a lab could commit to false data initially (mitigated by auditor evidence verification)

Edge Cases

- Clock synchronization: Timestamps are self-reported; malicious actors could backdate
- Network partitions: Mirrors could diverge temporarily during outages
- Key compromise: If a party's API key is stolen, unauthorized submissions possible until revoked

Scalability Constraints

- Current implementation uses JSON file storage (suitable for demo, not production)
- Hash chain verification is $O(n)$ —grows linearly with event count
- Merkle tree rebuilds on each new event (could be optimized with incremental updates)
- Single-server architecture (production would need distributed deployment)

Dual-Use Risks

Could this help bad actors?

Risk: A malicious lab could study this system to understand how verification works and design circumvention strategies.

Mitigation: The cryptographic primitives are well-established and already public. Our implementation doesn't reveal new attack vectors. The greater risk is NOT having verification infrastructure.

Risk: Anonymous whistleblowing could be abused to make false accusations.

Mitigation: Concerns trigger investigation, not automatic action. Auditors verify claims before resolution. Repeated false reports from same anonymous ID could be flagged.

Responsible Disclosure

No vulnerabilities in external systems were discovered during development. The system is designed for authorized use by labs, auditors, and regulators.

If vulnerabilities are discovered in production deployment, we recommend:

1. Private disclosure to system operators
2. 90-day remediation window
3. Public disclosure after fix or window expiration

Ethical Considerations

Privacy: We prioritized privacy-by-design. Whistleblower identities never enter the system. ZK proofs minimize information disclosure.

Access control: Full ledger access is restricted to relevant parties (labs, auditors, government) because content requires expertise to interpret. This is a deliberate design choice balancing transparency with practicality.

Enforcement: The system creates technical infrastructure but does not itself enforce compliance. Enforcement requires regulatory adoption. This is appropriate—technical systems should not replace democratic governance.

Future Improvements

1. **Database backend:** PostgreSQL/SQLite for production scale
2. **Distributed consensus:** Byzantine fault-tolerant mirroring across parties
3. **Hardware attestation:** Integration with TEEs for stronger evidence commitments
4. **Formal verification:** Prove cryptographic properties mathematically
5. **Incremental Merkle trees:** $O(\log n)$ updates instead of full rebuilds
6. **Audit logging for the audit log:** Meta-level tamper detection
7. **Integration APIs:** Connectors for existing lab infrastructure

Appendix B: AI/LLM Prompts Used

This project was developed with assistance from Claude (Anthropic). Key prompts included:

1. "Help me design a tamper-proof audit logging system for AI governance"
2. "Implement Merkle tree proof generation and verification in Python"
3. "Create a zero-knowledge proof system for proving event count thresholds"
4. "Design a multi-party mirroring system where tampering by any party is detectable"
5. "Implement client-side anonymous ID generation using Web Crypto API"
6. "Add role-based authentication with API keys to the FastAPI backend"
7. "Write comprehensive tests for the transparency ledger workflow"

The AI assisted with code implementation, architecture decisions, and documentation. All code was reviewed and tested by the human developer(s).

Appendix C: Repository Structure

```
ai-flight-recorder/
├── backend/
│   ├── api.py           # FastAPI REST endpoints (29 endpoints)
│   ├── audit_log.py     # Hash chain implementation
│   ├── auth.py          # Role-based authentication
│   ├── crypto_utils.py  # SHA-256 utilities, anonymous ID
│   ├── merkle_tree.py   # Merkle tree & proofs
│   ├── mirror_simulation.py # Multi-party mirroring demo
│   ├── models.py        # Pydantic data models
│   ├── transparency.py  # Transparency ledger
│   └── zk_proofs.py      # Zero-knowledge proofs
├── frontend/
│   └── app.py            # Streamlit dashboard
├── tests/               # 153 passing tests
├── data/                # Persistent storage
└── requirements.txt
```

GitHub: <https://github.com/zukapustikova/ai-governance-transparency-ledger>

Built for the International Technical AI Governance Hackathon