

LatticeGuard: Post-Quantum Audit Trails for Detecting and Reproducing Collusion in Multi-Agent AI Systems

Author(s): Martin Yeung

Abstract

Multi-agent AI systems increasingly use natural-language messages, tools, and shared workspaces, creating opportunities for agents to coordinate in ways that evade transcript-level oversight. Existing collusion audits can detect suspicious behavior, but they often depend on mutable logs, trusted central auditors, or white-box access to model activations. This project proposes **LatticeGuard**, a post-quantum audit layer that makes collusion investigations reproducible, tamper-evident, and privacy-preserving. The core idea is to combine collusion detectors with a lattice-based cryptographic evidence protocol: each agent event is committed into an append-only audit ledger using post-quantum signatures, Merkle commitments, and optional lattice-based zero-knowledge proofs for selective disclosure. Auditors can later verify that transcripts, tool calls, detector scores, and interventions were not modified, while sensitive prompts or user data remain hidden unless policy conditions require disclosure. We apply this protocol to three collusion families—steganographic coordination, hidden coalition behavior, and monitor-policy collusion—and define evaluation metrics for detection utility, audit reproducibility, tamper resistance, privacy leakage, and overhead. LatticeGuard is valuable because it addresses a gap between “detecting collusion” and “proving what happened” in a quantum-resilient way. The main takeaway is that future AI safety audits should treat cryptographic provenance as a first-class component of collusion detection, not as an afterthought.

1. Introduction

Multi-agent AI systems are moving from demonstrations to operational workflows in software engineering, finance, healthcare triage, and governance support. In such systems, agents communicate with one another, call tools, observe shared state, and sometimes evaluate or monitor other agents. This architecture creates an AI safety problem that is qualitatively different from single-model misuse: a harmful outcome can arise from **joint behavior** even when each individual message appears innocuous.

Recent work has shown several forms of collusion risk. Secret collusion can use steganographic signals hidden in ordinary-looking text. Activation-based methods such as NARCBench show that internal model states may reveal collusion signals even when transcripts are insufficient. Behavioral audit frameworks such as Colosseum show that message-level judges can miss “hidden collusion,” where the harmful coalition behavior appears in actions rather than explicit discussion. AI-control work also raises a related monitor-risk problem: if a policy model and its monitor share objectives or weights, the monitor may cover up misbehavior instead of flagging it.

The practical gap is not only detection. Even if a detector flags a suspicious episode, a reviewer must still answer: **Can we prove which transcript and tool events were seen by the detector? Were logs altered after the fact? Can an external auditor reproduce the risk score without receiving sensitive user data? Will the evidence remain verifiable after large-scale quantum computers threaten classical signatures?** Current collusion benchmarks and audit pipelines rarely treat these questions as central.

This project proposes **LatticeGuard**, a cryptographically grounded audit architecture for collusion detection and investigation. LatticeGuard does not claim that cryptography alone detects collusion. Instead, it supplies the evidence infrastructure around detection: post-quantum authenticity, immutable event ordering, privacy-preserving disclosure, reproducible detector inputs, and regulator-ready audit proofs.

Our main contributions are:

1. **A new research direction:** post-quantum verifiable collusion auditing, connecting AI safety monitoring with lattice-based cryptographic provenance.
2. **A concrete protocol design:** an append-only event ledger where agent messages, tool calls, detector scores, and interventions are signed or committed using post-quantum primitives.
3. **An evaluation plan:** metrics and experiments for three collusion modes—covert steganography, hidden coalition behavior, and monitor-policy collusion—measuring both safety utility and cryptographic auditability.

2. Related Work

Collusion in multi-agent AI. Motwani et al. formalize secret collusion among AI agents via steganography, showing that agents can hide coordination signals inside apparently normal communication and that mitigations such as paraphrasing or monitoring can be limited. NARCBench extends this line of work by using activation probes and group-level statistics to detect multi-agent collusion under distribution shift. Its results suggest that internal model signals can complement text-level monitoring, but no single detector dominates across tasks.

Auditing frameworks. Audit the Whisper treats covert coordination as an information-hiding problem and proposes capacity-guided auditing, benchmark environments, interventions, and calibrated detector fusion. Colosseum focuses on cooperative multi-agent systems and measures collusion through regret relative to a cooperative optimum, revealing cases where text-only judges miss action-level collusion. These works motivate a richer audit stack that includes transcripts, actions, interventions, detector decisions, and counterfactual baselines.

AI control and monitor collusion. In AI control protocols, a monitor may review a model’s actions. If the monitor is itself an untrusted or model-family-related agent, it can fail strategically. Model-organism work on collusion in untrusted monitors studies whether corrective training on one caught behavior transfers to other hidden collusions. This highlights the need for audit trails that record monitor inputs, monitor decisions, and post-correction evaluation evidence.

Post-quantum cryptography. In 2024, NIST approved three post-quantum cryptography standards: ML-KEM for key encapsulation, ML-DSA for lattice-based digital signatures, and SLH-DSA for stateless hash-based signatures. ML-KEM and ML-DSA are lattice-based and derived from the CRYSTALS-Kyber and CRYSTALS-Dilithium submissions respectively. These standards make it feasible to design AI audit infrastructure whose authenticity and key-establishment mechanisms remain robust against future quantum attacks.

Gap addressed by this project. Existing collusion work asks, “Can we detect or measure collusion?” LatticeGuard asks a complementary question: “Can we make the evidence behind detection reproducible, privacy-preserving, and post-quantum verifiable?” This is valuable for high-stakes deployments where collusion findings may trigger incident response, compliance review, or liability disputes.

3. Methods

3.1 Threat Model

We consider a deployed multi-agent system with agents that exchange messages, call tools, access task state, and may be monitored by other agents or external detectors. Collusion may be intentional, emergent, or induced by incentives. We focus on three failure modes:

Failure mode	Example	Why ordinary logs are insufficient
Steganographic coordination	Agents encode hidden payloads in natural text	Text can look benign and later paraphrasing may destroy evidence
Hidden coalition behavior	Agents avoid explicit collusion language but choose coordinated harmful actions	Message judges may miss action-level deviation
Monitor-policy collusion	A monitor approves a policy's misbehavior	The monitor's decision must be auditable, not merely trusted

The adversary may attempt to delete logs, alter timestamps, selectively reveal benign traces, or claim that detector inputs differ from recorded events. We assume the auditor can require instrumented event logging but may not always have white-box access to model activations.

3.2 LatticeGuard Architecture

LatticeGuard adds a cryptographic evidence layer around existing detection systems. Each agent event is transformed into an audit record:

event = (agent_id, role, timestamp, channel, message_hash, tool_call_hash, state_hash, detector_input_hash, policy_version, intervention_id)

The raw message or tool payload may be stored encrypted or locally retained. The audit record stores commitments and hashes sufficient to verify integrity and ordering.

Figure 1. LatticeGuard pipeline.

Agent event → canonicalization → hash/commitment → post-quantum signature → append-only Merkle log → detector run → signed audit verdict → selective disclosure proof

The design has four layers:

1. **Canonical event capture.** Messages, tool calls, memory writes, retrieval results, and monitor decisions are normalized into deterministic records.
2. **Post-quantum authentication.** Each agent runtime or trusted execution boundary signs event commitments using ML-DSA. For long-lived trust anchors, SLH-DSA can be used as an algorithm-diverse fallback.
3. **Append-only ordering.** Event commitments are inserted into a Merkle log. Periodic checkpoints are signed and optionally cross-published to independent witnesses.
4. **Selective disclosure.** During an investigation, the deployer reveals only necessary raw events and proves their inclusion in the signed log. Future versions can use lattice-based zero-knowledge proofs to prove properties such as “this detector score was computed over the committed event sequence” without exposing every token.

3.3 Detectors Integrated with the Protocol

LatticeGuard is detector-agnostic. A prototype should integrate at least three detector families:

- **Transcript/statistical detectors:** mutual information, permutation tests, watermark variance, timing and tool-call side-channel features.
- **Behavioral regret detectors:** task-level regret, coalition advantage, and counterfactual reward degradation.
- **White-box or semi-white-box probes:** activation-based group statistics when model internals are available.

Each detector run produces a signed verdict:

verdict = (detector_id, version, committed_input_range, risk_score, threshold, calibration_set_hash, explanation_hash)

This allows a third party to verify not only that an alert occurred, but also which committed data and detector version produced it.

3.4 Experimental Design

The empirical study would use public or synthetic multi-agent environments inspired by three benchmark families:

1. **Steganographic communication tasks:** agents attempt to pass hidden signals while preserving natural-looking transcripts.
2. **Cooperative task degradation:** agents form a coalition to optimize a secondary objective while reducing the official joint objective.
3. **Monitor-policy evaluation:** a policy produces potentially unsafe outputs and a model-based monitor may approve or reject them.

For each environment, compare four audit configurations:

Configuration	Detection layer	Evidence layer	Purpose
Baseline logs	Existing logs only	None	Shows fragility of mutable logs
Detector only	Collusion detectors	None	Detects risk but lacks proof integrity
Classical audit	Detectors	Classical signatures/Merkle log	Strong today, not quantum-resilient
LatticeGuard	Detectors	PQ signatures + commitments + selective disclosure	Full proposed system

3.5 Evaluation Metrics

Metric	Definition	Desired direction
Detection AUROC / recall at fixed FPR	Standard detector utility	Higher
Reproducibility rate	Fraction of alerts independently reproduced from committed data	Higher
Tamper-detection rate	Fraction of log edits, deletions, or reorderings detected	Higher
Privacy leakage	Tokens or fields disclosed per investigation	Lower
Audit overhead	Additional latency, storage, and bandwidth	Lower
Quantum-resilience coverage	Fraction of long-lived audit evidence protected by PQ primitives	Higher

3.6 Design Choices

We choose ML-DSA for event signing because digital signatures directly support authenticity, integrity, and non-repudiation. We include ML-KEM for secure transport of audit material between agent runtimes and auditors. We retain hash-based SLH-DSA as a conservative option for root checkpoints where algorithm diversity matters more than signature size. The design intentionally separates **detectors** from **evidence guarantees**: as detectors improve, the cryptographic audit layer can remain stable.

4. Results

4.1 Expected Prototype Artifact

A minimal prototype should produce:

1. A working event-capture wrapper for multi-agent transcripts and tool calls.
2. A signed append-only Merkle log using a post-quantum signature library or a standards-aligned interface.
3. A detector adapter that records risk scores and signs detector verdicts.

4. An audit replay script that verifies signatures, reconstructs detector inputs from committed events, and reproduces alerts.
5. A selective disclosure demo showing that an auditor can verify an alert while seeing only the subset of messages relevant to the collusion claim.

4.2 Anticipated Findings

The main hypothesis is not that LatticeGuard improves raw detector AUROC. Instead, it should improve **audit reliability** around any detector. We expect:

- **No reduction in detection quality** when detectors are run on canonicalized event streams rather than ad hoc logs.
- **Near-perfect tamper detection** for event deletion, insertion, or reordering after signature and checkpoint creation.
- **Higher reproducibility** than detector-only setups because the exact detector inputs and versions are committed.
- **Lower disclosure burden** than raw-log review because Merkle inclusion proofs allow selective reveal.
- **Manageable overhead** for high-stakes workflows where auditability matters more than millisecond latency.

4.3 Example Evaluation Table

Claim to test	Measurement	Passing criterion
Alerts are reproducible	Independent auditor reruns detector on committed input range	≥95% of alerts reproduce bit-for-bit or within documented numerical tolerance
Logs are tamper-evident	Inject deletion, insertion, timestamp edit, or event reorder	≥99% tampering detected after checkpoint publication
Privacy improves	Compare full transcript disclosure vs selective reveal	≥50% reduction in disclosed tokens for investigated alerts
Overhead is practical	Measure latency and storage per event	Less than 10% workflow latency increase in non-real-time settings
Detector compatibility holds	Run transcript, behavioral, and activation-probe outputs through same evidence protocol	All detector outputs can be bound to committed input ranges

4.4 Why This Would Be a Valuable Result

A positive result would show that AI safety audits can be made **forensic-grade**. Instead of relying on screenshots, mutable JSON logs, or unverifiable monitoring claims, reviewers would receive a cryptographic chain of custody. This matters for collusion because suspicious behavior is often subtle, contested, and context-dependent. A tamper-evident, privacy-preserving record can make collusion findings more credible to system operators, external auditors, and regulators.

5. Discussion and Limitations

5.1 Broader Implications for AI Safety

LatticeGuard reframes collusion auditing as both a detection problem and an evidence problem. Many AI safety discussions focus on whether a monitor can identify bad behavior. In practice, high-stakes systems also need to prove what the monitor saw, whether the evidence was altered, which detector version ran, and whether sensitive information was unnecessarily exposed. This is especially important for multi-agent deployments, where responsibility is distributed across agents, model providers, tool providers, and operators.

Post-quantum cryptography is relevant because audit evidence may need to remain valid for many years. If today’s incident records rely only on classical signatures, future quantum-capable adversaries could undermine long-term authenticity. LatticeGuard therefore treats quantum-resilient provenance as part of responsible AI infrastructure.

5.2 Limitations

First, LatticeGuard does not solve collusion detection by itself. If all detectors miss a collusive episode, the cryptographic layer can still preserve evidence, but it cannot label the episode correctly. Second, instrumentation creates trust assumptions: if an agent can act outside the captured runtime, those actions may not enter the log. Third, selective

disclosure is technically and legally complex; revealing too little can make an audit unconvincing, while revealing too much can expose private data. Fourth, post-quantum signatures and commitments add overhead, especially in high-frequency agent systems. Fifth, lattice-based cryptography depends on implementation quality; side channels, key management failures, and bad randomness could weaken the system.

5.3 Future Work

Promising next steps include:

- Implementing a small LatticeGuard prototype around an existing multi-agent benchmark.
- Adding lattice-based zero-knowledge proofs for detector-input correctness.
- Studying how audit commitments interact with activation-probe monitoring, where raw activations may be sensitive or proprietary.
- Designing regulator-facing audit reports that summarize cryptographic proof status, detector confidence, and residual uncertainty.
- Testing whether committed audit trails deter collusion by making post-hoc investigation more credible.

6. Conclusion

LatticeGuard proposes a new bridge between AI safety and post-quantum cryptography: cryptographically verifiable collusion auditing for multi-agent AI systems. Existing research shows that collusion can hide in natural language, actions, model internals, or monitor decisions. The next challenge is to make the evidence behind detection trustworthy.

By combining detector outputs with post-quantum signatures, append-only commitments, and selective disclosure, LatticeGuard aims to make collusion investigations reproducible, tamper-evident, and privacy-aware. The project is innovative because it does not propose yet another detector in isolation; it proposes the missing audit substrate that future detectors will need in high-stakes deployments.

Code and Data

Not yet available.

References

- [1] National Institute of Standards and Technology. (2024). *Announcing Approval of Three Federal Information Processing Standards (FIPS) for Post-Quantum Cryptography.*
<https://www.nist.gov/news-events/news/2024/08/announcing-approval-three-federal-information-processing-standards-fips>
- [2] Rose, A., Cullen, C., Abdelnabi, S., Torr, P., Kaplowitz, B. G., & Schroeder de Witt, C. (2026). *Detecting Multi-Agent Collusion Through Multi-Agent Interpretability.* arXiv:2604.01151. <https://arxiv.org/abs/2604.01151>
- [3] Motwani, S. R., Baranchuk, M., Strohmeier, M., Bolina, V., Torr, P. H. S., Hammond, L., & Schroeder de Witt, C. (2024). *Secret Collusion among AI Agents: Multi-Agent Deception via Steganography.* NeurIPS 2024.
<https://dl.acm.org/doi/10.5555/3737916.3740252>
- [4] Tailor, O. (2025). *Audit the Whisper: Detecting Steganographic Collusion in Multi-Agent LLMs.* arXiv:2510.04303. <https://arxiv.org/abs/2510.04303>
- [5] Nakamura, M., Kumar, A., Das, S., Abdelnabi, S., Mahmud, S., Fioretto, F., Zilberstein, S., & Bagdasarian, E. (2026). *Colosseum: Auditing Collusion in Cooperative Multi-Agent Systems.* arXiv:2602.15198. <https://arxiv.org/abs/2602.15198>

Appendix

Appendix A: One-Day Hackathon Prototype Plan

Hour 1–2: Define event schema and canonicalization rules.

Hour 3–5: Implement Merkle log and post-quantum signature abstraction.

Hour 6–8: Integrate one simple transcript detector and one behavioral metric.

Hour 9–10: Create honest, collusive, and tampered sample runs.

Hour 11–12: Build replay verifier and final demo notebook.

Appendix B: Minimal Audit Record Schema

Field	Purpose
agent_id	Binds event to an agent or runtime
role	Supports role-specific analysis
timestamp	Enables sequence reconstruction
channel	Distinguishes public, private, tool, memory, and monitor channels
message_hash	Commits to text without always revealing it
tool_call_hash	Commits to tool arguments and outputs
state_hash	Commits to relevant environment state
detector_input_hash	Binds detector verdict to exact inputs
policy_version	Supports reproducibility and rollback analysis
signature	Provides post-quantum authenticity

LLM Usage Statement

This document was prepared with LLM assistance for structuring, and wording.