
Blind Audit¹

Giles Edkins
Trajectory Labs

Owen Walker
Independent

With
Apart Research

Abstract

The project presents an AI Governance Challenge Framework utilizing Trusted Execution Environments (TEEs) to address the critical problem of proving a dataset's safety without revealing the private data itself. The core approach involves a protocol where a Data Provider, holding a private dataset, executes a Challenger's detection script (packaged as a WebAssembly module) inside a TEE. This process generates a cryptographically signed attestation package, which proves the script was run correctly over the full dataset. A Judge component (LLM-powered) verifies this attestation and examines any flagged documents for unsafe content. The hackathon demo is deployed on a cloud-based server utilizing real TEE hardware, establishing a verifiable mechanism for dataset safety assurance.

1. Introduction

We are addressing a fundamental challenge of accountability and trust in AI systems: the need to cryptographically prove that a private training dataset is safe (i.e., free of harmful or sensitive content) to an external party without exposing the data itself. In our example datasets we have

¹ Research conducted at the [Technical AI Governance Challenge](#), 2026

focused on (mocked) PII, regulated, and sensitive information; however the principle behind this protocol extends to many kinds of potentially unsafe or unwelcome data.

The primary failure mode we are targeting is a scenario where a malicious or compromised Data Provider attempts to attest to the safety of a dataset that secretly contains unsafe content.

Our main contributions are:

1. A verifiable protocol for running dataset challenges within a Trusted Execution Environment (TEE), specifically AWS Nitro Enclaves.
2. The definition of a secure Attestation Package, including TEE-signed results and a Merkle root of the dataset for integrity checks.
3. The integration of a Judge (LLM-powered) to verify attestations and evaluate content flagged by challenge modules.

2. Related Work

Our work builds upon existing cryptographic primitives like Merkle trees for data integrity, secure computing concepts such as Trusted Execution Environments (TEEs/Intel SGX/AWS Nitro Enclaves), and the use of WebAssembly for sandboxed execution of challenge logic. It differs from traditional auditing by offering a protocol that combines the privacy guarantees of TEEs with the public verifiability of a challenge-response system. This addresses the critical gap where current systems either require full data access for an audit or rely on unverifiable claims from the data holder.

3. Methods

The framework operates as a challenge-response protocol between a Data Provider, a Challenger, and a Judge. The Data Provider hosts the private dataset and an execution environment. Challengers submit detection scripts compiled to WebAssembly (WASM) binaries. The key design choices are:

- **Trusted Execution Environment (TEE):** The WASM challenge module is run inside a TEE (Nitro Enclave) to ensure the Data Provider cannot tamper with the script or the dataset during execution.
- **WASM Challenge Modules:** These binaries read a single document (as a string of bytes) and return a boolean indicating safe or unsafe.
- **Verifiable Results (Attestation Package):** Upon completion, the TEE generates a cryptographically signed Attestation Package. The quote within the package includes the WASM module's hash, the dataset's Merkle root, and the result's hash, providing proof of execution integrity, signed with a TEE key.

- **Judge Verification:** The Judge service verifies the TEE signature chain (Demo Root CA -> Attestation Authority -> TEE Signing Key) and uses an LLM (requires an Anthropic or OpenAI API key) to evaluate the content of any flagged document.

4. Results

The primary result is the successful demonstration of the complete challenge-response protocol in a real hardware TEE environment. The framework successfully achieved the following objectives:

- **Verifiable Execution:** Executed a WebAssembly (WASM)-based challenge module (e.g., a PII detector) on a sample dataset within a Trusted Execution Environment (TEE).
- **Attestation Generation:** Generated a cryptographically signed Attestation Package upon completion. This package contains the TEE quote, the dataset's Merkle root, and the result's hash, providing proof of execution integrity.
- **Judge Verification:** Demonstrated the Judge service's ability to verify the TEE signature chain and use an integrated LLM (when configured with an API key) to evaluate the content of any flagged document.

5. Discussion and Limitations

The primary implication of this framework for AI safety is the establishment of a protocol for verifiable, confidential computing in dataset governance. It allows for independent verification of dataset safety properties without exposing the raw data, thereby resolving the conflict between privacy and auditability. This is a crucial step for deploying trustworthy models in sensitive industries.

Limitations and Future Work

- The protocol assumes that the attested Merkle root of the dataset is actually useful for something - this implies the need for a counterpart protocol for establishing the validity of ML training runs against a given dataset, identified by its Merkle root. This is not demonstrated in our work but would be an exciting future direction.
- The judge is assumed to be trusted by all parties to correctly assess the selected data.
- The judge is assumed to be trusted by the data provider not to leak the small amount of data that it is given. Even small amounts of data revealed this way might reveal information about a data provider's proprietary "secret sauce".
- The challenger is assumed to be able to write a script to detect bad data without being able to see any information about the dataset itself. Future extensions to this protocol might allow more information to be (selectively) released to the challenger.

6. Conclusion

Code and Data

- **Code repository:** <https://github.com/edkins/blind-audit>

Author Contributions (optional)

Giles Edkins developed:

- Infrastructure and AWS Deployment scripts
- Data provider document checking and attestation logic
- Hardware TEE deployment (on AWS Nitro Enclave)

Owen Walker developed:

- Zero Knowledge set membership logic
- Challenger submission page
- Judge prompting

References

Appendix (*optional*)

Supplementary material such as additional figures, detailed methodology, prompts used, extended results, etc.

LLM Usage Statement

We used Claude and Google Antigravity to develop a lot of the code.

Template version 2.0 — January 2026