

# Engram Provenance: Substrate-Rooted Provenance for Safety-Critical AI Systems

Aure Ecker-Fils

2026

## **Abstract**

This paper introduces Engram Provenance, a substrate-rooted provenance model for AI systems. Building on the Engram Layer and the Engram Signature, provenance is defined as continuity of execution-realized identity across time, perturbation, and substrate variation. Engram Provenance formalizes lineage as a sequence of stable Engram Signatures and provides minimal conditions for auditability, chain-of-custody, and substrate-rooted trust in safety-critical domains.

# Contents

<b>1</b>	<b>Abstract</b>	<b>4</b>
<b>2</b>	<b>Introduction</b>	<b>4</b>
<b>3</b>	<b>Background</b>	<b>4</b>
<b>4</b>	<b>Substrate Triplet and Execution Space</b>	<b>4</b>
<b>5</b>	<b>Constructive Example of a Candidate <math>\Phi</math></b>	<b>5</b>
<b>6</b>	<b>Provenance and Lineage</b>	<b>6</b>
6.1	Provenance Definition . . . . .	6
6.2	Continuity Condition . . . . .	6
6.3	Provenance Graph . . . . .	7
<b>7</b>	<b>Cryptographic Binding</b>	<b>7</b>
7.1	Deterministic Key Derivation . . . . .	7
7.2	Stability-Uniqueness Tension . . . . .	7
7.3	Signed Outputs . . . . .	7
7.4	Verification . . . . .	7
<b>8</b>	<b>Chain-of-Custody</b>	<b>8</b>
8.1	Inference Provenance . . . . .	8
8.2	Training Provenance . . . . .	8
8.3	Fine-Tuning Lineage . . . . .	8
8.4	Hardware and Runtime Transitions . . . . .	8
<b>9</b>	<b>Applications</b>	<b>8</b>
9.1	Medicine . . . . .	8
9.2	Justice . . . . .	8
9.3	Defense and Robotics . . . . .	8
9.4	Multi-Agent Systems . . . . .	8
<b>10</b>	<b>Threat Model</b>	<b>9</b>
10.1	Spoofing . . . . .	9
10.2	Drift . . . . .	9
10.3	Hardware Replacement . . . . .	9
10.4	Adversarial Considerations: Difficulty of Spoofing $\Phi$ . . . . .	9
<b>11</b>	<b>Limitations</b>	<b>9</b>
<b>12</b>	<b>Conclusion</b>	<b>10</b>
<b>13</b>	<b>Comparison with Existing Provenance and Attestation Approaches</b>	<b>10</b>
13.1	Metadata and Pipeline Provenance . . . . .	10
13.2	Model Watermarking and Fingerprinting . . . . .	10
13.3	Reproducible and Deterministic ML . . . . .	11
13.4	Trusted Execution Environments (TEEs) and Hardware Roots of Trust . . . . .	11

13.5	Hardware Physical Unclonable Functions (PUFs) . . . . .	11
13.6	Summary of Distinction . . . . .	11
14	Security Model and Considerations	12
15	Design Principles	13
16	Limitations and Future Work	14
17	Conclusion and Outlook	15
18	Appendix A: Illustrative Experimental Sketch for Engram Signature Extraction	16
18.1	A.1 Experimental Objective . . . . .	16
18.2	A.2 Experimental Setup . . . . .	16
18.3	A.3 Feature Collection . . . . .	17
18.4	A.4 Signature Construction . . . . .	17
18.5	A.5 Stability and Distance Evaluation . . . . .	18
18.6	A.6 Toward Cryptographic Use . . . . .	18
18.7	A.7 Limitations of the Sketch . . . . .	18
18.8	A.8 Expected Empirical Failure Modes . . . . .	19
19	References	19

# 1 Abstract

This paper introduces Engram Provenance, a substrate-rooted provenance model for AI systems. Building on the Engram Layer and the Engram Signature, provenance is defined as continuity of execution-realized identity across time, perturbation, and substrate variation. Engram Provenance formalizes lineage as a sequence of stable Engram Signatures and provides minimal conditions for auditability, chain-of-custody, and substrate-rooted trust in safety-critical domains.

## 2 Introduction

Provenance in AI systems is typically implemented through metadata, checkpoints, and external identifiers [6], [7]. These mechanisms do not capture execution-realized identity and cannot track continuity across runtime variation, hardware drift, or heterogeneous execution [1], [2], [4]. As modern AI systems increasingly operate in safety-critical contexts, provenance must reflect the substrate that realizes cognition rather than the artifacts that describe it. Engram Provenance builds on the Engram Signature [13] by treating identity as a persistent structural pattern and lineage as the continuity of this pattern across executions. This yields a substrate-rooted provenance model suitable for audit, verification, and chain-of-custody in safety-critical domains.

## 3 Background

The Engram Layer characterizes the execution substrate through which trained functions become realized functions during inference [14]. Numerical stability and floating-point perturbation can influence execution pathways [3], [4], [5]. Runtime variation has been shown to induce structured divergence in model behavior [1], [2]. The Engram Signature formalizes identity as a cross-layer structural pattern stable under bounded perturbations [13]. Substrate-rooted identity has precedent in hardware security, where silicon variation enables physically unclonable functions [12]. Engram Provenance generalizes this principle to execution-level variation in AI systems.

## 4 Substrate Triplet and Execution Space

Let

$$\mathcal{E} = (H, E, M)$$

denote the execution substrate, where

$$H$$

is the hardware,

$$E$$

the runtime engine, and

$$M$$

the model.

Let

$$\mathcal{X}(\mathcal{E})$$

be the space of executions of

$$\mathcal{E}$$

.

Define a mapping

$$\Phi : \mathcal{X}(\mathcal{E}) \rightarrow \mathcal{S}$$

from executions to a space of cross-layer structural patterns

$$\mathcal{S}$$

.

Let

$$\mathcal{X}_\varepsilon(\mathcal{E}) \subseteq \mathcal{X}(\mathcal{E})$$

denote executions under perturbations bounded by

$$\varepsilon$$

.

The Engram Signature is the equivalence class

$$\text{ES}(\mathcal{E}) = [\Phi(x)]_{x \in \mathcal{X}_\varepsilon(\mathcal{E})}$$

.

## 5 Constructive Example of a Candidate $\Phi$

A plausible instantiation of

$$\Phi$$

illustrates feasibility. Let each execution

$$x$$

produce a tuple of measurable cross-layer features:

$$f(x) = (\tau(x), \mu(x), \rho(x), \alpha(x))$$

where:

•

$$\tau(x)$$

: timing-derived microarchitectural signals [4],[5]

•

$$\mu(x)$$

: memory-hierarchy access patterns [4],[5]

•

$$\rho(x)$$

: floating-point perturbation response under controlled noise [3]

•

$$\alpha(x)$$

: activation-space structural summaries (e.g., layerwise activation-covariance spectra, sparsity distributions, or attention-entropy profiles)

Define

$$\Phi(x) = h(f(x))$$

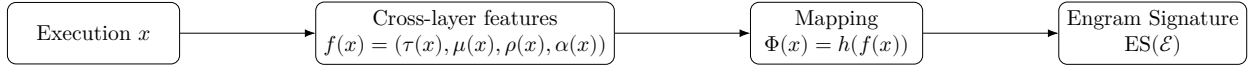
where

$$h$$

is a dimensionality-reducing transformation mapping the feature tuple into

$$\mathcal{S}$$

This construction is illustrative rather than prescriptive.



**Diagram: Execution – Features –  $\Phi$  – Engram Signature**

## 6 Provenance and Lineage

### 6.1 Provenance Definition

Provenance is the ordered sequence of Engram Signatures observed across executions of a system:

$$(\text{ES}(\mathcal{E}_t))_{t=0}^T$$

### 6.2 Continuity Condition

A sequence forms a lineage if:

$$d(\text{ES}(\mathcal{E}_t), \text{ES}(\mathcal{E}_{t+1})) \leq \delta$$

for all

$$t$$

•

A practical choice for

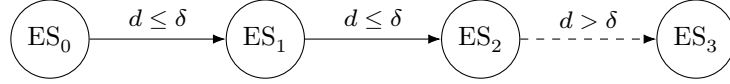
$$d$$

may be cosine distance in the reduced signature space or a Hamming distance after quantization, though the framework does not depend on any specific metric.

### 6.3 Provenance Graph

A provenance graph is defined by:

- **Nodes:** Engram Signatures
- **Edges:** Execution transitions
- **Labels:** Substrate-level invariants



**Diagram: Provenance Graph**

## 7 Cryptographic Binding

### 7.1 Deterministic Key Derivation

Let

$$\text{ES}(\mathcal{E})$$

be the Engram Signature.

Define:

$$k_{\text{priv}} = \text{KDF}(\text{ES}(\mathcal{E})), \quad k_{\text{pub}} = \text{PK}(k_{\text{priv}})$$

The private key is never stored; it is re-derived whenever needed, following the construction introduced in the Engram Signature [13].

### 7.2 Stability-Uniqueness Tension

A practical construction must address the tension between stability and uniqueness. Engram Signatures must remain stable under bounded perturbations yet sufficiently distinct across substrates. This mirrors the classical challenge in PUF systems[12], where noisy but device-specific measurements require fuzzy extractors or error-correcting mechanisms to produce stable keys. Engram-derived keys would require analogous constructions to ensure reproducibility and collision resistance [8].

### 7.3 Signed Outputs

Inference outputs can be signed using

$$k_{\text{priv}}$$

, binding them to the realized system.

### 7.4 Verification

Verification requires only signature stability, not hardware retention.

## 8 Chain-of-Custody

### 8.1 Inference Provenance

Each output is associated with the Engram Signature of the execution that produced it.

### 8.2 Training Provenance

Training provenance refers to the Engram Signature of a specific realized training run, not the abstract training procedure of a model family. Let:

$$\mathcal{E}_{\text{train}}^{(i)} = (H_{\text{train}}, E_{\text{train}}, M_{\text{init}})$$

denote training run

$i$

. The corresponding signature:

$$\text{ES}(\mathcal{E}_{\text{train}}^{(i)})$$

captures the execution-realized identity of that run. Engram Provenance does not assume stability across different training runs; it records each independently.

### 8.3 Fine-Tuning Lineage

Continuity across fine-tuning cycles is established when successive signatures satisfy the continuity condition.

### 8.4 Hardware and Runtime Transitions

Lineage is preserved when transitions maintain signature stability; otherwise, a new lineage begins.

## 9 Applications

### 9.1 Medicine

Diagnostic outputs can be tied to the realized system that produced them.

### 9.2 Justice

AI-generated evidence can maintain chain-of-custody across time and hardware.

### 9.3 Defense and Robotics

Autonomous agents require persistent identity across updates and drift.

### 9.4 Multi-Agent Systems

Substrate-rooted identity enables verifiable inter-agent communication.



## 10 Threat Model

### 10.1 Spoofing

Cross-layer emergence complicates direct spoofing of Engram Signatures.

### 10.2 Drift

Large perturbations may break continuity and terminate lineage.

### 10.3 Hardware Replacement

Identity continuity may not survive substrate discontinuity.

### 10.4 Adversarial Considerations: Difficulty of Spoofing $\Phi$

Spoofing

$\Phi$

requires reproducing the cross-layer structural pattern extracted from the execution substrate. An attacker must approximate the joint distribution of execution-level features across hardware, runtime, and model layers.

- **Emulation of execution statistics** requires reproducing microarchitectural timing and scheduling behavior.
- **Replay attacks** assume trace determinism, which does not hold for equivalence-class-based signatures [2].
- **Distillation-based approximation** reproduces functional behavior but not execution-level variation tied to hardware and runtime.
- **Model extraction combined with hardware-profiling attacks** [5] may approximate the model’s functional surface while probing cache behavior, timing characteristics, or other microarchitectural signals. However, reproducing the *joint* cross-layer structural pattern required to spoof

$\Phi$

remains difficult due to coupling between activation-space structure, perturbation response, and hardware-dependent execution features.

These considerations do not establish cryptographic hardness but indicate that spoofing requires reproducing multi-layer physical and computational properties. Practical deployments would require fuzzy extractors or error-correcting constructions.

## 11 Limitations

Engram Provenance depends on signature stability, runtime instrumentation, and controlled perturbation bounds. Hardware discontinuity and major runtime changes may break lineage. Cryptographic constructions require noise-tolerant mechanisms analogous to those used in PUF systems.

## 12 Conclusion

Engram Provenance extends the Engram Signature from identity to continuity, providing a substrate-rooted provenance model for safety-critical AI systems. By grounding provenance in execution-realized identity, it establishes a basis for auditability, chain-of-custody, and trust in modern AI deployments.

## 13 Comparison with Existing Provenance and Attestation Approaches

Engram Provenance differs from existing provenance, watermarking, and attestation mechanisms in both **where identity is rooted** and **what continuity is meant to capture**. This section situates the framework relative to established approaches and clarifies its distinct contribution.

### 13.1 Metadata and Pipeline Provenance

Conventional AI provenance systems track external artifacts [6],[7] such as:

- dataset hashes
- training logs
- model checkpoints
- version-controlled pipelines

These mechanisms establish **procedural lineage**, but they do not describe how a model is *realized* at inference time. Two systems with identical metadata may diverge in execution due to hardware, runtime, or numerical variation.

Engram Provenance complements metadata-based approaches by grounding identity in **execution-level structural behavior**, not descriptive records.

### 13.2 Model Watermarking and Fingerprinting

Watermarking techniques embed identifiable patterns [8],[9] into weights or outputs to prove ownership or origin. They:

- require intentional modification of the model
- may be removable or forgeable
- track model artifacts rather than runtime realization

Engram Provenance does not embed identifiers. Identity emerges from the **interaction of hardware, runtime, and model** during execution. Watermarking marks *what the model is*; Engram Provenance characterizes *how the model is realized*.

The two are orthogonal.

### 13.3 Reproducible and Deterministic ML

Reproducibility efforts aim to eliminate execution variability [1],[2],[3] through strict control of seeds, libraries, and hardware. Their goal is **identical outputs** across runs.

Engram Provenance instead treats bounded execution variability as **informative signal**. Structured variation across hardware and runtime becomes the basis for a stable, substrate-rooted identity.

Reproducibility seeks sameness; Engram Provenance seeks **continuity under controlled variation**.

### 13.4 Trusted Execution Environments (TEEs) and Hardware Roots of Trust

TEEs, TPMs, and secure enclaves anchor trust in **stored secrets** and **attested firmware states**. Identity is declarative: it reflects what the hardware claims about itself.

Engram Provenance differs in two key respects:

1. **No stored root secret** — cryptographic material is derived from execution-level structural patterns, not provisioned keys.
2. **Behavioral attestation** — trust is tied to how the system behaves under execution, not solely to firmware measurements.

TEEs attest software-state integrity; Engram Provenance attests **execution-realized identity**. They are complementary layers.

### 13.5 Hardware Physical Unclonable Functions (PUFs)

PUFs derive identity from microscopic manufacturing variation [12]. Engram Provenance generalizes this idea from **static physical variation** to **dynamic cross-layer execution behavior**.

Property	PUFs	Engram Provenance
Source of identity	Physical manufacturing variation	Execution-level structural variation
Measurement	Electrical challenge-response	Cross-layer execution features
Stability mechanism	Fuzzy extractors	Fuzzy extractors / noise-tolerant schemes
Scope	Single device	Hardware-runtime-model triplet

Engram Provenance can be viewed as a **computational analogue of PUFs** at the level of AI system execution.

### 13.6 Summary of Distinction

Engram Provenance is unique in that it:

- roots identity in **execution behavior**, not metadata or embedded markers

- treats bounded variability as **signal**, not noise to be eliminated
- enables continuity tracking across time and controlled substrate drift
- derives cryptographic trust from **measured system behavior**, not stored secrets

These properties position Engram Provenance as a **complementary trust layer** alongside existing provenance, watermarking, and attestation technologies.

## 14 Security Model and Considerations

**14.0.0.1 Adversary Capabilities** The adversary may observe, perturb, or partially influence execution behavior. This includes: - inducing bounded numerical variation, - modifying runtime conditions such as scheduling, memory locality, or hardware state, - attempting to force divergent execution pathways through crafted inputs, - observing structural patterns leaked through timing, cache behavior, or signature-adjacent metadata.

The adversary is assumed to have no ability to extract or clone the underlying substrate-rooted identity of the system.

**14.0.0.2 Adversary Limitations** The adversary cannot: - replicate the physical substrate or its perturbation-stable structural patterns, - forge Engram Signatures without access to the realized execution substrate, - induce unbounded perturbations without detection, - bypass the measurement boundary defined by the Engram Layer.

These limitations reflect the substrate-rooted nature of identity and the bounded-perturbation stability of Engram Signatures.

**14.0.0.3 Security Goal** The security objective is continuity of execution-realized identity across time, perturbation, and substrate variation.

A system satisfies this objective when its Engram Signature remains within the permitted stability region across executions, enabling: - provenance continuity, - lineage reconstruction, - substrate-rooted trust.

**14.0.0.4 Out-of-Scope Attacks** The following are explicitly out of scope: - physical substrate replacement, - invasive hardware attacks, - microarchitectural compromise that destroys the stability region, - full-system compromise that invalidates the Engram Layer.

These attacks break the foundational assumptions of substrate-rooted identity.

**14.0.0.5 Sensitivity to Measurement Noise** Engram Signatures tolerate bounded perturbations but remain sensitive to: - extreme numerical instability, - pathological hardware drift, - adversarially induced chaotic execution regimes.

Systems must ensure that measurement noise remains within the stability region defined by the Engram Layer.

**14.0.0.6 Exposure of Structural Information** Structural leakage through timing, cache behavior, or execution traces may reveal partial information about the Engram Signature.

However, such leakage does not enable cloning of the substrate-rooted identity, as the signature is tied to the physical realization of execution.

**14.0.0.7 Adversarial Manipulation of Execution Behavior** Adversaries may attempt to force execution into unstable regions.

Mitigations include: - perturbation-bounded inference, - deterministic execution modes, - runtime monitoring of signature drift.

These mechanisms ensure that deviations beyond the stability region are detectable.

**14.0.0.8 Interaction with Existing Trust Anchors** Engram Provenance complements, rather than replaces, existing trust anchors such as: - TEEs, - attestation protocols, - cryptographic identity systems.

Substrate-rooted identity provides an additional layer of assurance by binding provenance to the realized execution substrate rather than metadata or external identifiers.

**14.0.0.9 Cryptographic Binding and Key Derivation** Deterministic key derivation from Engram Signatures enables: - substrate-rooted cryptographic identity, - binding of lineage to cryptographic artifacts, - secure chain-of-custody.

Keys derived from Engram Signatures inherit the stability properties of the underlying substrate-rooted identity.

## 15 Design Principles

Engram Provenance is guided by a set of design principles that shape how identity, continuity, and trust are derived from the realized execution of an AI system. These principles articulate the architectural commitments of the framework and distinguish it from metadata-based or secret-based approaches.

**15.0.0.1 Substrate-Rooted Identity** Identity is anchored in the **hardware–runtime–model triplet** rather than in descriptive metadata or embedded markers. The Engram Signature captures structural properties of how a computation is realized on a specific substrate, making identity emergent rather than provisioned.

**15.0.0.2 Bounded Variability as Signal** Execution variability arising from scheduling, numerical perturbation, and microarchitectural effects is treated as **informative structure**, not noise to be eliminated. Engram Provenance leverages this bounded variability to characterize the substrate and distinguish it from others.

**15.0.0.3 Cross-Layer Feature Coupling** No single feature class (timing, memory, perturbation response, activation structure) is assumed sufficient. Identity emerges from the **joint distribution** of cross-layer features, making spoofing or replication significantly harder than matching any individual signal.

**15.0.0.4 Stability Under Perturbation** Repeated executions on the same substrate should yield Engram Signatures within a stability bound. This principle ensures that identity is **robust to benign drift** in thermal conditions, scheduling, and numerical noise while remaining sensitive to substrate changes.

**15.0.0.5 Continuity Over Time** Provenance is defined as **continuity of Engram Signatures** across executions. This enables lineage tracking, drift detection, and chain-of-custody reasoning without requiring immutable hardware identifiers or persistent secrets.

**15.0.0.6 Behavioral Attestation** Trust is derived from **measured execution behavior**, not from stored secrets, firmware claims, or declarative attestations. This aligns Engram Provenance with behavioral security models and complements hardware-rooted trust mechanisms.

**15.0.0.7 Non-Intrusive Measurement** Feature extraction should rely on profiling mechanisms that do not materially alter the execution path. This principle minimizes observer effects and preserves the fidelity of the Engram Signature.

**15.0.0.8 Compatibility and Composability** Engram Provenance is designed to coexist with existing provenance pipelines, watermarking schemes, reproducibility frameworks, and TEEs. It provides an **additional trust layer** rather than replacing established mechanisms.

## 16 Limitations and Future Work

Engram Provenance establishes a conceptual and formal foundation for execution-realized identity, but several limitations remain. These limitations highlight open research questions and opportunities for empirical validation.

**16.0.0.1 Hardware Diversity and Generalization** The framework assumes that different substrates produce measurably distinct execution-level signatures. While preliminary sketches suggest feasibility, large-scale validation across heterogeneous CPUs, GPUs, accelerators, and cloud runtimes is required. Future work should quantify separability across architectures, manufacturing generations, and thermal/power regimes.

**16.0.0.2 Runtime Sensitivity and Environmental Drift** Execution-level features may be influenced by background load, power-management policies, thermal throttling, or virtualization layers [4],[5]. These factors may introduce variance beyond the assumed perturbation bound. Future work should explore adaptive perturbation models, dynamic stability thresholds, and runtime-aware normalization strategies.

**16.0.0.3 Feature Selection and Representation Learning** The choice of cross-layer features and the transformation

$$h$$

strongly affects signature quality. The current formulation is intentionally flexible, but systematic evaluation is needed to identify robust feature sets, projection methods, and quantization schemes. Learned representations (e.g., contrastive encoders over execution traces) may offer improved stability and separability.

**16.0.0.4 Adversarial Robustness and Mimicry Resistance** While reproducing the full joint distribution of timing, memory, perturbation response, and activation structure is difficult, adversarial strategies remain an open area of study. Future work should evaluate mimicry attacks, model-extraction-plus-profiling attacks, and adaptive adversaries capable of shaping execution behavior. Formalizing mimicry difficulty is an important direction.

**16.0.0.5 Integration with Existing Attestation Pipelines** Engram Provenance is designed to complement TEEs, watermarking, and metadata-based provenance. Practical deployments must address how Engram Signatures interact with existing trust anchors, how conflicts between layers are resolved, and how hybrid attestation protocols can combine behavioral and declarative evidence.

**16.0.0.6 Measurement Overhead and Deployment Constraints** Feature extraction introduces computational and instrumentation overhead. Real-world deployments must balance measurement fidelity with latency, throughput, and energy constraints. Lightweight profiling techniques, incremental signature updates, and sampling-based measurement strategies are promising directions.

**16.0.0.7 Formal Security Guarantees** Engram Provenance does not claim cryptographic hardness. Future work may explore formal models of adversarial capability, bounds on mimicry difficulty, and connections to computational PUFs, behavioral attestation, and information-theoretic stability. Establishing formal guarantees remains an open research challenge.

**16.0.0.8 Longitudinal Behavior and Drift Modeling** The continuity condition assumes bounded drift over time, but long-term hardware aging, driver updates, and runtime evolution may alter execution characteristics. Future work should investigate longitudinal signature stability, drift-aware lineage models, and mechanisms for controlled re-enrollment.

## 17 Conclusion and Outlook

Engram Provenance introduces a substrate-rooted approach to identity, continuity, and trust for safety-critical AI systems. By grounding provenance in execution-level structural behavior rather than metadata, embedded identifiers, or stored secrets, the framework reframes identity as an emergent property of the hardware–runtime–model triplet. The Engram Layer and Engram Signature formalize this perspective, enabling continuity tracking across executions, detection of substrate drift, and behavioral attestation without requiring invasive instrumentation or deterministic execution.

The framework complements existing provenance pipelines, watermarking techniques, reproducibility efforts, and hardware attestation mechanisms by providing a behavioral trust layer that captures how a system is realized during inference. While Engram Provenance does not claim cryptographic hardness, it establishes a principled foundation for deriving stable, noise-tolerant identity from bounded execution variability. The accompanying experimental sketch demonstrates that the required features are measurable with current tooling, and the limitations outlined above highlight a rich space for empirical validation, adversarial analysis, and integration with existing trust anchors.

As AI systems increasingly operate across heterogeneous hardware, distributed runtimes, and long operational lifetimes, execution-realized identity becomes a critical missing layer in the provenance and attestation landscape. Engram Provenance offers an initial architecture for this layer, providing

a conceptual substrate on which future systems, protocols, and security mechanisms can be built. The work opens a path toward a broader research agenda in behavioral attestation, computational PUFs, and cross-layer identity for AI systems—an agenda that will require collaboration across machine learning, systems, hardware security, and formal methods.

## 18 Appendix A: Illustrative Experimental Sketch for Engram Signature Extraction

This appendix outlines a minimal experimental setup demonstrating how a candidate mapping

$$\Phi$$

could be instantiated and measured in practice. The goal is not to prescribe a definitive construction, but to show feasibility using currently available tooling.

### 18.1 A.1 Experimental Objective

The experiment evaluates whether execution-level features collected from repeated runs of the same model under bounded perturbations exhibit:

1. **Intra-substrate stability** — low variance across executions on the same substrate.
2. **Inter-substrate separability** — measurable divergence across different substrates.

These properties correspond to the stability and uniqueness requirements of Engram Signatures.

### 18.2 A.2 Experimental Setup

**18.2.0.1 Model** A small transformer-based language model (e.g., 125M–350M parameters) is used to ensure tractable instrumentation. The model remains fixed across all trials.

**18.2.0.2 Runtime Engine** Inference is performed using a deterministic, instrumentable runtime (e.g., CPU or GPU backend with profiling hooks enabled). Controlled perturbations are introduced via:

- randomized but bounded thread scheduling
- injected floating-point noise within machine epsilon
- minor thermal or frequency variation within safe limits

These define the bounded execution space

$$\mathcal{X}_\epsilon$$

.

**18.2.0.3 Hardware** Experiments are conducted across at least three distinct substrates, for example:

- two different CPU microarchitectures
- one GPU platform

Each hardware–runtime combination defines a distinct execution substrate

$$\mathcal{E}$$

.



## 18.3 A.3 Feature Collection

For each execution

$x$

, the following cross-layer features are collected.

### 18.3.0.1 Timing Signals

- per-layer latency distributions
- variance in kernel or operator execution time
- cache-miss-correlated timing fluctuations (if accessible)

These capture microarchitectural influences on execution.

### 18.3.0.2 Memory Access Patterns

- hardware performance counters (cache misses, bandwidth usage)
- page-level access frequency histograms

These characterize the memory-hierarchy footprint of the model’s execution.

**18.3.0.3 Perturbation Response** The model is evaluated under small, controlled floating-point perturbations. Measured features include:

- divergence in intermediate activation norms
- sensitivity of final logits to perturbation

This estimates numerical stability characteristics of the execution path.

**18.3.0.4 Activation Structural Summaries** Rather than logging raw activations, compact structural descriptors are computed, such as:

- layerwise activation-covariance spectra (e.g., top singular values)
- sparsity ratios per layer
- attention-entropy statistics

These capture global structural properties of the realized computation while remaining low-dimensional.

## 18.4 A.4 Signature Construction

All features are concatenated into a vector:

$$f(x) = (\tau(x), \mu(x), \rho(x), \alpha(x))$$

A dimensionality-reducing transformation

$h$

(e.g., PCA followed by quantization or a learned projection) maps

$$f(x)$$

into a compact representation:

$$\Phi(x) = h(f(x))$$

The Engram Signature

$$\text{ES}(\mathcal{E})$$

is defined as the equivalence class of

$$\Phi(x)$$

over repeated executions within the perturbation bound.

## 18.5 A.5 Stability and Distance Evaluation

For each substrate:

1. Run

$$N$$

repeated executions under bounded perturbations.

2. Compute pairwise distances

$$d(\Phi(x_i), \Phi(x_j))$$

within the same substrate.

3. Compare with distances across different substrates.

Candidate distance metrics include cosine distance in the reduced space or Hamming distance after quantization.

Desired outcome:

$$\text{Intra-substrate distance} \ll \text{Inter-substrate distance}$$

This demonstrates feasibility of stability-bounded identity.

## 18.6 A.6 Toward Cryptographic Use

If the reduced signatures show:

- low intra-substrate variance, and
- sufficient inter-substrate separation,

then a fuzzy extractor or error-correcting scheme (as used in PUF systems) could derive stable cryptographic keys from noisy signature measurements.

## 18.7 A.7 Limitations of the Sketch

This setup:

- does not establish optimal feature choices

- does not prove spoofing resistance
- serves only as a feasibility demonstration

A full implementation would require adversarial testing, larger model scales, and broader hardware diversity.

## 18.8 A.8 Expected Empirical Failure Modes

Several empirical failure modes may arise in practice, including:

1. **Insufficient inter-substrate separation** if hardware-dependent effects are weak or underrepresented.
2. **Excessive intra-substrate variance** due to thermal drift, load, or nondeterministic scheduling.
3. **Over-dominance of model-level features** that obscure substrate differences.
4. **Measurement intrusiveness**, where profiling perturbs execution.
5. **Adversarial feature mimicry**, reducing inter-substrate distance.
6. **Dimensionality-reduction artifacts** that collapse distinct substrates.

These failure modes highlight that Engram Signature extraction is an empirical systems problem rather than a guaranteed invariant.

## 19 References

- [1] Y. Simmhan, B. Plale, and D. Gannon, “A survey of data provenance in e-science,” *IEEE Internet Comput.*, vol. 9, no. 5, pp. 34–41, 2005.
- [2] P. Buneman, S. Khanna, and W. C. Tan, “Why and where: A characterization of data provenance,” in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 2001.
- [3] L. Bouthillier, K. Khetan, C. Vincent, and P. Vincent, “On the reproducibility of neural network predictions,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021.
- [4] H. Pham, Z. Dai, Q. V. Le, and L. J. Li, “The unreasonable ineffectiveness of determinism in deep learning,” in *Proc. Conf. Mach. Learn. Syst. (MLSys)*, 2020.
- [5] X. Mei and X. Chu, “Dissecting GPU memory hierarchy through microbenchmarking,” in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, 2016.
- [6] A. Ecker-Fils, “Engram Signature: Substrate-Rooted Identity in AI Systems,” *Zenodo*, 2026.
- [7] A. Ecker-Fils, “Engram: Execution-Realized Cognition in Modern AI Systems — A Unified Framework for the Execution Substrate and Its Falsification Criteria,” *Zenodo*, 2026.
- [8] D. Goldberg, “What every computer scientist should know about floating-point arithmetic,” *ACM Comput. Surv.*, vol. 23, no. 1, pp. 5–48, 1991.

- [9] Q. Ge, Y. Yarom, D. Cock, and G. Heiser, “A survey of microarchitectural side-channel vulnerabilities, attacks, and defenses in cryptography,” *Proc. IEEE*, vol. 106, no. 1, pp. 168–186, 2018.
- [10] R. Maes, “Physically unclonable functions: A study on the state of the art and future research directions,” *Proc. IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.
- [11] Y. Adi, C. Baum, M. Cisse, B. Pinkas, and J. Keshet, “Turning your weakness into a strength: Watermarking deep neural networks by backdooring,” in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2018.
- [12] H. Chen, B. Li, Y. Li, and S. Song, “DeepMarks: A digital fingerprinting framework for deep neural networks,” *arXiv:1804.03648*, 2018.
- [13] V. Costan and S. Devadas, “Intel SGX explained,” *IACR Cryptol. ePrint Arch.*, 2016.
- [14] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, “Security and privacy challenges in industrial Internet of Things,” *ACM Comput. Surv.*, vol. 49, no. 4, pp. 1–31, 2015.