Model Identity Attestations in the Enterprise Identity Stack

A Technical Brief

Fall Risk Research | March 2026
anthony@fallrisk.ai | https://fallrisk.ai

Purpose

This document shows how structural model identity — the ability to verify which neural network is computing inside a deployed system — composes with existing enterprise identity standards. It is grounded in real measurements from six open-weights language models executed inside an NVIDIA H100 Confidential Computing enclave.

The document is a practical companion to the Fall Risk Research paper series (available at https://fallrisk.ai). It assumes familiarity with OAuth 2.0, SPIFFE/SPIRE, and JWT-based identity flows. This brief focuses on structural model identity in environments where privileged runtime measurement is available. The broader evidence framework also includes API-accessible evidence classes for narrower provenance and class-membership claims.

The Problem

Enterprise identity standards solve three problems well:

- **Who is authorized?** OAuth 2.0/2.1 manages delegated authorization.
- **Which workload is running?** SPIFFE/SPIRE issues cryptographic identities to software workloads.
- **Who was provisioned?** SCIM manages identity lifecycle across systems.

They share a common assumption: that the model inside the agent is known. If a model is substituted, distilled from an unauthorized source, or silently updated, every credential, token, and authorization grant remains valid. The credential authenticates the deployment. It does not authenticate the model.

Model identity verification addresses the missing layer: confirming which neural network is actually executing, independent of its deployment credentials.

What a Model Identity Measurement Produces

A structural identity measurement runs inside a Trusted Execution Environment (Intel TDX + NVIDIA Confidential Computing). The model processes a standardized set of challenge inputs, and the measurement engine observes the model's internal computation to produce a compact numerical fingerprint — a 64-dimensional identity vector that distinguishes the specific model in the validated regime.

The measurement produces an **evidence bundle** containing:

The fingerprint cannot be used to reconstruct the model's weights, architecture, or training data. It functions as an identifier, not a disclosure.

Empirical Validation

The measurement demonstrated in this document was performed on March 17, 2026, inside a GCP Confidential VM (a3-highgpu-1g, Intel TDX, NVIDIA H100 80GB in CC mode).

Across the six-model dataset (1,536 total measurements), the measurement process produced zero execution failures and no observed fingerprint collisions. Models from the same family (Qwen-0.5B vs Qwen-1.5B, Llama-1B vs Llama-3B) produce entirely distinct fingerprints. Structural identity is individual, not familial.

Field	Description
`model_id`	Operator-provided model identifier
`fingerprint`	64-dimensional structural identity vector
`weight_hash`	SHA-256 of the model's weight files
`bind_root`	Cryptographic binding linking fingerprint to attestation session
`verifier_nonce`	Freshness guarantee — prevents replay
`gpu_nonce`	SHA-256(bind_root) — binds GPU attestation to the session
`measured_at`	Timestamp of measurement
`engine_ver`	Version of the measurement engine
`seeds`	Random seeds used for challenge selection
`challenge_set_hash`	Hash of the prompt bank used
`tdx_attestation`	Intel TDX attestation token (CPU enclave proof)
`gpu_attestation`	NVIDIA CC attestation token (GPU enclave proof)

Model	Family	Parameters	Structural Identity (mean)	Fingerprint Digest
Qwen-2.5-0.5B	Qwen	0.5B	0.134	[recorded]
Qwen-2.5-1.5B	Qwen	1.5B	0.133	[recorded]
Llama-3.2-1B	Llama	1B	0.116	[recorded]
Llama-3.2-3B	Llama	3B	0.116	[recorded]
Gemma-2-2B	Gemma	2B	0.359	[recorded]
Mistral-7B	Mistral	7B	0.105	[recorded]

Carrying Model Identity in Existing Tokens

The evidence bundle is the raw material. It does not belong inside a token — it is too large, too sensitive, and too complex for runtime authorization flows. Instead, a compact **model identity claim** is derived from the bundle and carried as a namespaced claim inside standard JWT-based tokens.

Claim Schema: `fallrisk.ai/model_identity`

```
{
  "ver": "1.0",
  "measurement_type": "structural",
  "evidence_ref": "https://fallrisk.ai/evidence/[bundle-id]",
  "bundle_digest": "[SHA-256 of canonical evidence bundle]",
  "fingerprint_digest": "[SHA-256 of 64D fingerprint vector]",
  "bind_root": "[redacted in public examples]",
  "weight_hash": "[redacted in public examples]",
  "attestation_digest": "[SHA-256 over TDX+GPU attestation materials]",
  "measured_at": "2026-03-17T08:43:15Z",
  "evidence_fresh_until": "2026-03-24T08:43:15Z",
  "engine_ver": "itpuf_v1.2_patched",
  "match_status": "enrolled_match",
  "trust_mode": "tee_backed",
  "policy_scope": "structural-identity-verification-v1"
}
```

Key design decisions:

- **Digests, not raw values.** The token carries SHA-256 digests of the fingerprint and the full evidence bundle — not the raw vectors or attestation bodies. This preserves verifiability without disclosing the model's identity geometry.
- **Evidence reference, not evidence.** The `evidence_ref` field provides a URI to the full stored bundle for relying parties that require deeper audit. The `bundle_digest` allows the relying party to verify the referenced bundle hasn't been modified.
- **Evidence freshness separate from token lifetime.** The `evidence_fresh_until` field tracks when the measurement evidence itself becomes stale, independent of the JWT `exp` field. A token can expire (requiring re-issuance) while the underlying evidence remains fresh, or the evidence can become stale while the token is still valid. The relying party checks both.
- **Domain-qualified claim name.** A domain-qualified claim name (`fallrisk.ai/model_identity`) avoids collisions with standard claims or other vendors' custom fields.

JWT Access Token Example (RFC 9068 Profile)

Based on the real Qwen-0.5B measurement:

```
{
  "sub": "model:qwen-2.5-0.5b-instruct:enrolled",
  "iss": "fallrisk-attestation-service",
  "aud": "enterprise-gateway-demo",
  "jti": "a0c3c174-63d4-4db0-a681-8f2b96a18872",
  "iat": 1773736995,
  "exp": 1773823395,
  "cnf": {
    "jkt": "[presenter-key-thumbprint]"
  },
  "fallrisk.ai/model_identity": {
    "ver": "1.0",
    "measurement_type": "structural",
    "evidence_ref": "https://fallrisk.ai/evidence/[bundle-id]",
    "bundle_digest": "[SHA-256 -- illustrative]",
    "fingerprint_digest": "[SHA-256 -- illustrative]",
    "bind_root": "[redacted]",
    "weight_hash": "[redacted]",
    "attestation_digest": "[redacted]",
    "measured_at": "2026-03-17T08:43:15Z",
    "evidence_fresh_until": "2026-03-24T08:43:15Z",
    "engine_ver": "itpuf_v1.2_patched",
    "match_status": "enrolled_match",
    "trust_mode": "tee_backed",
    "policy_scope": "structural-identity-verification-v1"
  }
}
```

Standard JWT fields (`sub`, `iss`, `aud`, `jti`, `iat`, `exp`, `cnf`) follow RFC 9068. The model identity claim is an additional namespaced payload that a model-identity-aware relying party reads; standard OAuth infrastructure passes it through unchanged.

JWT-SVID Example (SPIFFE Profile)

The same claim carried in a SPIFFE JWT-SVID, for workload-to-workload identity at Layer 7:

```
{
  "sub": "spiffe://fallrisk.ai/model/qwen-2.5-0.5b-instruct",
  "iss": "fallrisk-attestation-service",
  "aud": "spiffe://enterprise.example/gateway",
  "jti": "a0c3c174-63d4-4db0-a681-8f2b96a18872",
  "iat": 1773736995,
  "exp": 1773823395,
  "cnf": {
    "jkt": "[presenter-key-thumbprint]"
  },
  "fallrisk.ai/model_identity": { "...same claim block..." }
}
```

The only change is in `sub` (SPIFFE URI instead of model identifier) and `aud` (SPIFFE trust domain instead of application identifier). The model identity claim is identical.

Relying-Party Verification Flow

When a relying party receives a token containing the `fallrisk.ai/model_identity` claim:

Step 1: Standard JWT verification. Validate the token signature against the issuer's published keys. This confirms that the token was signed by a trusted issuer accepted by the relying party. (Standard OAuth/SPIFFE infrastructure.)

Step 2: Standard token checks. Verify `exp` (token not expired), `aud` (token intended for this relying party), `iat` (token not from the future). (Standard JWT validation.)

Step 3: Presenter binding. If the token includes a `cnf` claim, verify proof-of-possession — that the entity presenting the token holds the private key corresponding to the thumbprint. This prevents token theft and replay. (RFC 7800.)

Step 4: Evidence freshness check. Read `evidence_fresh_until` from the model identity claim. If the current time exceeds this value, the underlying measurement evidence is stale even if the token is still valid. The relying party may reject, require re-measurement, or apply a degraded trust policy.

Step 5: Model identity policy check. Read `match_status` and `policy_scope`. Apply local policy: `enrolled_match` + acceptable `policy_scope` proceeds; `no_match` or unrecognized scope denies or escalates; unacceptable `trust_mode` restricts.

Step 6 (optional): Full evidence audit. Dereference `evidence_ref`. Download the full evidence bundle. Verify `bundle_digest` matches the digest in the token. Inspect the TDX and GPU attestation tokens independently. Verify `bind_root` consistency across CPU and GPU attestations.

Step 7: Access decision.

Outcome	Action
All checks pass	Allow the requested operation
Evidence stale but token valid	Restrict to low-sensitivity operations; request re-measurement
Model identity mismatch	Deny and alert
Attestation invalid	Deny and escalate

Security Properties

The composition of model identity attestations with standard authorization tokens must satisfy specific security properties. Each property is explicitly stated, classified, and traced to its source.

Four properties are formally **PROVED** in supplementary Coq files accompanying Paper 9 in the research series. Three are **TRACED** to existing cryptographic standards or prior formal work. One is

Property	Statement	Status	Source
Unforgeability	Cannot mint a valid model-identity claim without a genuine measurement in a genuine enclave	TRACED	Formally verified impossibility results (Coq) + TEE attestation chain
Non-replayability	A claim from one session cannot be replayed for a different session	TRACED	bind_root freshness + verifier nonce + JWT `jti`
Non-separability	The model-identity claim cannot be detached from its authorization context and presented in a different context without independent re-attestation	PROVED	ComposableIdentity.v A3 — 8 theorems, 6 axioms, 0 Admitted
Verification soundness	If RP verification passes, the model was genuinely attested by the claimed enclave	TRACED	TDX attestation + GPU attestation + bind_root binding
Issuer authenticity	Accepted token traces to an authorized issuer via the specific signing key	PROVED	IssuerAuthenticity.v T1 — 2 theorems, 3 axioms, 0 Admitted
Reference integrity	If the bundle retrieved at audit time differs from the bundle bound at issuance time, the digest check fails	PROVED	ReferenceIntegrity.v T1+T3 — 3 theorems, 2 axioms, 0 Admitted
Claim minimality	The token reveals only digests and references, not raw fingerprints or attestation internals	DESIGN	Token schema and redaction policy
Temporal validity	Any verification that checks only token expiry without checking evidence freshness inherits stale-evidence vulnerability	PROVED	ComposableIdentity.v B4 — temporal binding necessity
Audience binding	The token is scoped to a specific relying party	IMPLEMENTED	JWT `aud` claim (RFC 7519)
Evidence-class correctness	The claim declares its evidence class; RP checks claim type against required class	TRACED	Formal admissibility condition (Coq) + `measurement_type` field

IMPLEMENTED by standard JWT mechanisms. Two are **DESIGN**-constrained by the token schema. No security property of the composition is left implicit or unnamed.

What This Does Not Replace

Model identity verification is a complementary layer, not a replacement for any existing standard.

Standard	What It Does	What Model Identity Adds
OAuth 2.0/2.1	Manages delegated authorization	Verifies the model behind the authorized agent
SPIFFE/SPIRE	Issues workload identities	Verifies the model inside the identified workload
SCIM	Manages identity lifecycle	Provides the runtime evidence that lifecycle records reference
NGAC	Enforces attribute-based access control	Supplies the model-identity attribute for access decisions

In a typical enterprise deployment, SCIM is a natural place to carry inventory-side model identity data: which model is expected in which deployment, what evidence class is required, and when re-measurement is due. The JWT/SVID carries the runtime proof; SCIM carries the inventory and policy context.

Limitations

Attestation SDK availability. The TDX and GPU attestation tokens shown in the certificates reference validated attestation chains from prior empirical work (6 models, 1,536 measurements, 0 failures). Production deployment requires the Intel Trust Authority SDK and NVIDIA attestation SDK, whose availability and API stability are platform-dependent.

Evidence freshness is a policy decision. The `evidence_fresh_until` field provides a mechanism, not a recommendation. How long a measurement remains trustworthy depends on the operator's deployment cadence, the sensitivity of the operation, and whether the model is expected to change. The framework provides the plumbing; the organization owns the policy.

Structural identity only. This system verifies which model is running. It does not verify model quality, safety, fairness, accuracy, or fitness for any purpose. It answers "which model is this?" — not "is this model good?"

Privileged measurement required for structural claims. The integration described in this brief requires runtime access to the model's internal computation inside a trusted enclave. Where privileged runtime measurement is unavailable, the broader framework supports narrower provenance and class-membership claims through API-accessible evidence, as described in Papers 2 and 4 of the research series.

References

- Fall Risk Research paper series: https://fallrisk.ai
- "Composable Model Identity: Formal Hardening of Structural Attestations in the Enterprise Identity Stack" — DOI: 10.5281/zenodo.19099911
- "What Counts as Proof? Admissible Evidence for Neural Network Identity Claims" — DOI: 10.5281/zenodo.19058540
- RFC 9068: JWT Profile for OAuth 2.0 Access Tokens
- RFC 7519: JSON Web Tokens (JWT)
- RFC 7800: Proof-of-Possession Key Semantics for JWTs
- RFC 7515: JSON Web Signature (JWS)
- SPIFFE JWT-SVID specification: https://spiffe.io/docs/latest/spiffe-specs/jwt-svid/

Anthony Ray Coslett
Fall Risk Research
anthony@fallrisk.ai
https://fallrisk.ai