Raziel

NUS National University of Singapore | FinTech Lab School of Computing

Proof-carrying authorization for AI agents

Research evidence and a design-partnership invitation

Prof. Jungpil Hahn (Advisor)
Gita Alekhya Paul
NUS FinTech Lab, National University of Singapore

17 November 2026

Useful agents need bounded access to data

We built Raziel so another service can check an agent's permission independently.

An owner approves the access rules. Raziel proves that the rules and committed relationship facts authorize a grant for an exact agent, task, and data scope.

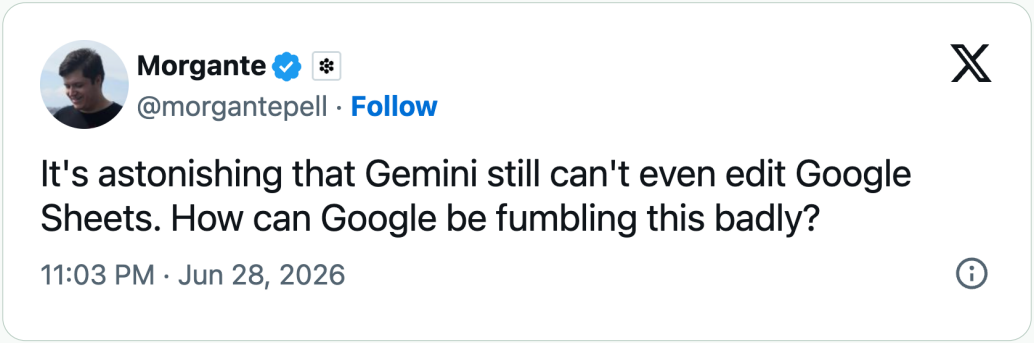
The owner retains the private relationship evidence. An independent verifier checks the proof. A gateway checks each signed request before releasing data.

What we demonstrated: two agents read separate synthetic ledgers under separate grants and return a settlement match with linked receipts.

Our next collaboration: a bounded partner workflow with agreed access rules, a sandbox dataset, and measurable admission and rejection tests.

Final report, executive summary and Chapters 4, 6–8.

Tool access leaves an authorization question



This public post about spreadsheet editing illustrates a practical concern: tool capability alone does not define which action an agent should take.

Illustrative motivation from the linked public post, June 2026.

Business problem

Trust has to scale beyond the pilot

In a pilot
Use test data, restrict tools, and supervise a small set of tasks.

In production
Agents act on live records across teams and systems. The permissions can differ for each task.

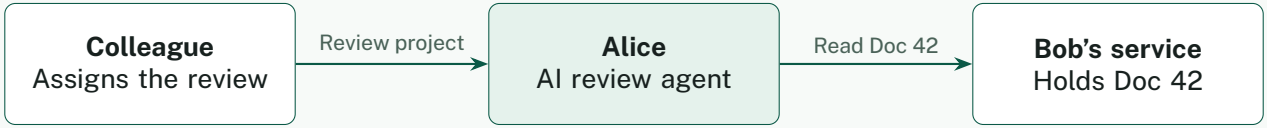
In McKinsey's 2025 global survey, 39% of respondents reported experimenting with agents and 23% reported scaling an agent system in at least one function. In finance, MAS and industry partners plan to test runtime safeguards through industry pilots and sandboxes.

An isolated environment limits what an agent can reach. A live system has to check what the agent is allowed to do.

McKinsey Global Survey, 2025 · MAS industry announcement, July 2026

Alice needs to read Doc 42

Alice is an AI agent asked to review a project. Doc 42 is restricted.



Approved rule

Only assigned reviewers can read Doc 42. Alice has no reviewer assignment.

A genuine signature can carry the wrong grant

In this baseline, Bob relies on the issuer to apply the reviewer rule.



Bob checks: issuer signature, read scope, Alice's key binding, and freshness. All pass. The grant still contradicts the reviewer rule.

Baseline: OAuth JWT access token with DPoP key binding.

Raziel withholds Doc 42 without a positive proof

Bob now requires an independently verified `ALLOW` proof for Alice's exact read.

The proof must match the approved policy and committed reviewer assignments.

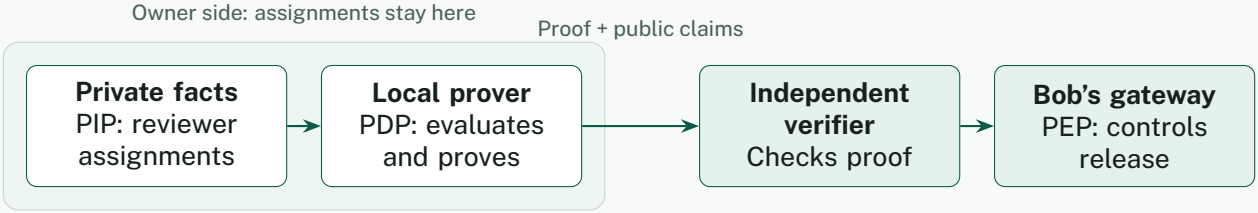


No assignment, no positive proof, no release.

Raziel blocks the read.

What stays private and what Bob still trusts

Bob checks the proof for an allowed read. The owner keeps the reviewer assignments.

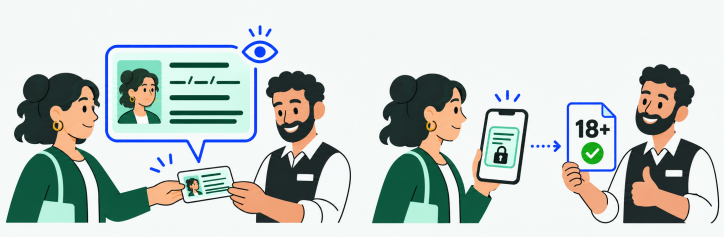


Trust still required

Policy and snapshot authorities, the approved proof verification key, and the gateway that enforces the release decision.

What a bar learns from an age check

A visitor can prove an age claim and keep the supporting evidence private.



Conventional full-ID check
The visitor shows a full ID. The bar checks the age threshold and sees the visitor's name, full date of birth, photo, and other printed details.

A zero-knowledge age check
The visitor's wallet proves "18 or over" from a trusted credential. The bar verifies the age claim. The name, birth date, and photo stay private.

Age-check analogy. The bar trusts the credential issuer and proof system.

Raziel applies this idea to permission evidence

	Age-check analogy	Raziel authorization
Approved check	Is this person 18 or over?	Does the approved policy allow this exact agent's task and scope?
Private evidence	The certified date of birth	Relationship facts, a satisfying path, and private conditions
Evidence to verify	A proof of the age claim	A proof that the committed inputs evaluate to `ALLOW`
Trusted inputs	The credential issuer	Approved policy and signed relationship snapshot

The verifier sees the approved policy and public proof claims. The relationship witness stays with the owner.

Each later data request still passes signature, scope, expiry, and replay checks.

Final report, Sections 4.1–4.6. The age check explains the privacy principle.

SAFR: Four components of runtime governance

Safeguards for Agentic Finance at Runtime

Raziel's focus: approve the rule, verify the read, record the decision.

Monetary Authority of Singapore

Agent Identity Foundation

Agents sign requests with registered keys.

Controls Repository

The registry stores approved access rules and their versions.

Disposition Engine

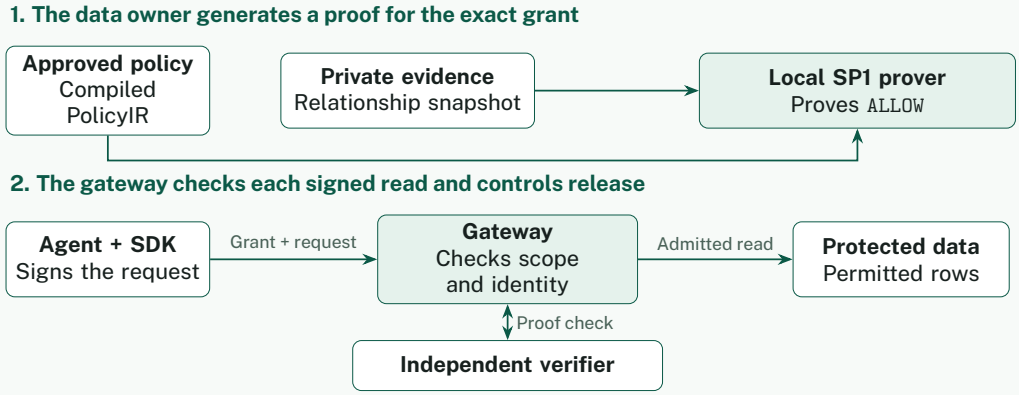
The gateway checks proof-backed authority before releasing records.

Audit Log

Protected-query receipts link the decision to the request and capability.

SAFR white paper, v1.1, October 2026, p. 8, Fig. 1

Raziel separates proof issuance from protected reads

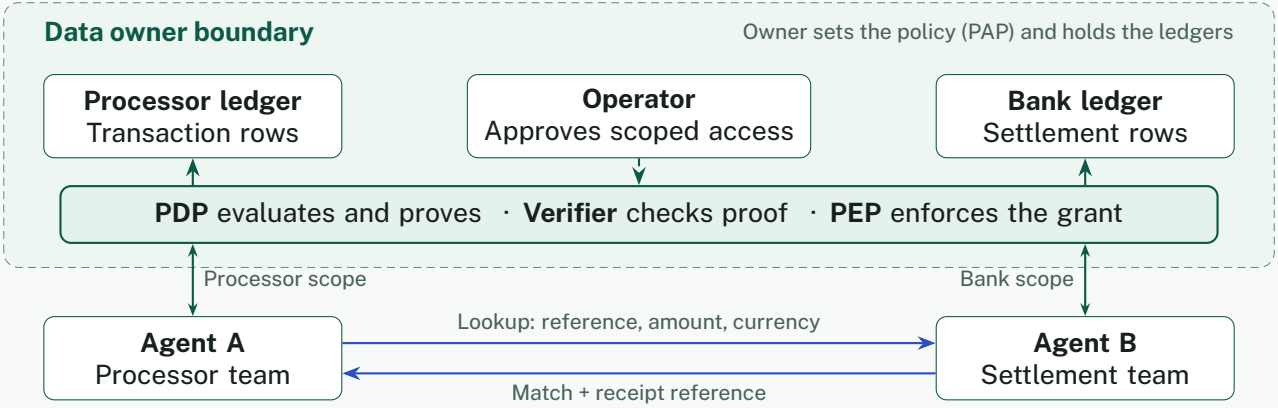


The verifier checks the proof against its approved policy registry. The gateway can then reuse the verified grant. Every read keeps authentication, scope, validity, replay, and dataset-binding checks.

Final report, Chapters 4–5.

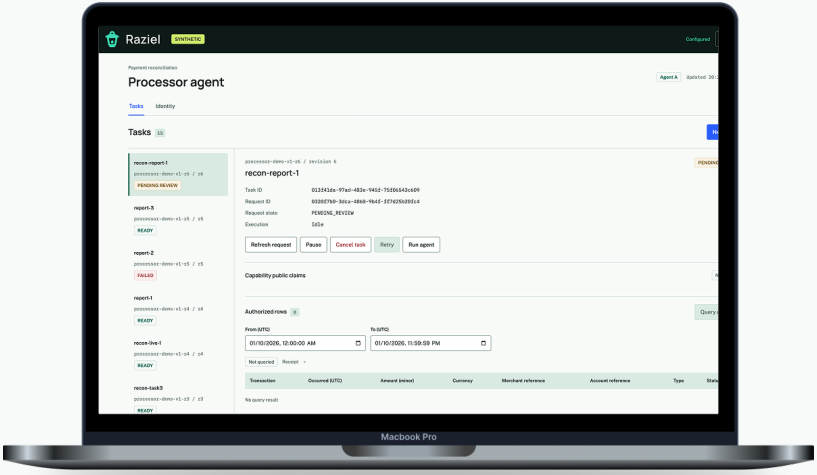
Two teams reconcile one payment

Each agent reads its own team's ledger to reconcile the payment.



Synthetic demo: each agent uses its own grant. The agents exchange lookup and match evidence.

Demo: two agents reconcile one payment



Original 2:35 recording

Approve scoped access
The operator reviews each request.

Read separate ledgers
Each agent uses its own grant.

Return a settlement match
Agent B sends the evidence to Agent A.

Watch the original demonstration.

The demonstration returns evidence from both ledgers



Final report, Section 6.4. The reconciliation tool compares authorized records.

Observed synthetic case

Agent A read three authorized processor records.

Agent B read bank records under a separate grant and returned a matching SGD 125.00 settlement with zero amount difference.

Both protected reads produced receipts linking the request to its agent, task, policy, and dataset revision.

We started with private data retrieval

Research direction	What it addresses	What we learned
Searchable encryption	Find matching records in encrypted data.	Queries and access patterns can still reveal information.
Attribute-based encryption (ABE)	Decrypt content with a key whose attributes satisfy its policy.	Separate the decision to grant access from the mechanism that protects content.
Computation on encrypted data	Evaluate a program on protected inputs.	Decide which inputs must stay private and which outputs another party needs to check.

Our pivot: finding a relevant record and having permission to read it are separate decisions.

Li et al. (2023), Zhang et al. (2019), Gui et al. (2023), Bethencourt et al. (2007), Goldwasser et al. (2013).

We compared ways to express authority

Authorization model	Basis for access	What it showed us
Role-based (RBAC)	Permissions assigned to roles.	Model duties through role assignments.
Attribute-based (ABAC)	Rules over attributes of the request and its context.	Express conditions beyond a role.
Private ABAC	An interactive check over encrypted attributes.	Kerl et al. showed how to evaluate rules over private inputs.
Relationship-based (ReBAC)	Tuples and usersets connect subjects to resources.	Zanzibar and SpiceDB gave us relationship rules and revision-aware checks.
Graph-policy systems	Rules over nodes, edges, or paths.	XACML4G separates graph policy from enforcement.

First and Second Interim Reports. Kerl et al. (2025), Pang et al. (2019), Mohamed et al. (2023, 2024).

We asked what the verifier could check independently

Approach	What the verifier checks	Design choice for Raziel
ReBAC decision API	A permission result from the relationship engine.	Keep the relationship rules. Add evidence that the approved rule allowed the read.
Restricted tokens and proof of possession	A scoped credential or proof that its presenter holds the bound key.	Use a reusable grant and check a fresh signature on each request.
Proof-carrying authorization	A logical derivation justifies the decision.	Keep independent checking without sending the private relationship path.
Verifiable computation	A proof binds a program's execution to public outputs.	Run the policy evaluator in a fixed zkVM guest and use PLONK to keep the witness private.

Pang et al. (2019), Birgisson et al. (2014), RFC 9449, Appel and Felten (1999), Parno et al. (2013), Gabizon et al. (2019).

Raziel attaches proof to a scoped grant

We built a reusable capability that carries independently checkable authorization evidence.

Issue	Evaluate an approved policy against committed relationships. Bind the grant to one agent key, task, action, and scope.
Verify	Check a local PLONK proof through a separate verifier. Keep relationship evidence with the owner.
Reuse	Admit later signed queries under the same grant. Check scope, expiry, and replay on every request.

We built and evaluated this design on a synthetic reconciliation workflow.

What we measured

Generate a proof once. Reuse the grant for signed reads within scope.

447 ms
One independent proof check

~16 min
One local proof generated

8/8
Tampered inputs rejected

21/21
Policy decisions matched SpiceDB

In the demo: Two agents used separate grants to read separate ledgers. They shared a settlement result and a linked receipt.

Separate proving and verification runs, 25–26 August 2026. Policy equivalence fixture suite.

A partner evaluation starts with one restricted action

We seek a workflow owner to define a bounded evaluation with us.

	Partner workflow	Doc 42 example
Agent	Which agent receives the task?	Alice, the project review agent
Action	What exact read or action is requested?	Read restricted Doc 42
Rule	What facts must justify permission?	Alice is an assigned reviewer

We propose one permitted request and one request the gateway must block. Together, we define the expected evidence and the acceptance criteria.

Proposed collaboration. The existing prototype evaluates protected data reads.

Design partners supply the workflow and test boundary

Partner contribution	Purpose in the proposed evaluation
Workflow and policy owner	Name the agent, resource, task, and access rules. Review the allowed and blocked cases.
Bounded data environment	Provide an approved sandbox or synthetic dataset with recorded provenance and permitted use.
Integration contact	Map the protected API, signing identity, and gateway boundary. Agree what the verifier can trust.

We bring: the Raziel prototype, policy and proof pipeline, agent SDK, independent verifier, and an evidence plan for the chosen workflow.

Design-partner scope proposed for discussion. Project information: raziel.gpaul.cc.

Sponsorship for the next evaluation

We seek research support tied to a defined partner evaluation.

Requested support

Evaluation compute and infrastructure

Proof engineering and integration effort

Access to domain and security reviewers

Proposed research outputs

A measured record of issuance and verification cost

An assessed allowed/blocked workflow and failure analysis

A reusable integration and evidence report

Start a scoping discussion through raziel.gpaul.cc. Funding, timing, data access, and publication terms will be agreed together.

Technical detail and research evidence

The following pages explain the design and the measured evidence.

- Research questions, proof statement, and SP1 guest inputs
- Policy roles, privacy boundaries, and implementation architecture
- Individual proving runs, independent verification, and negative controls
- Capability reuse, cost interpretation, and Hypha research directions

Three questions guide the evaluation

	Research question	Evidence sought
RQ1	Can a supported relationship-policy fragment retain its meaning and produce local proofs?	Semantic agreement and proving cost
RQ2	Can a verifier check the grant without receiving the relationship witness?	Public-field review, isolated verification, and mutation rejection
RQ3	Can agents reuse a grant and retain checks on every request?	Protected queries on both cold and cached paths

Evaluation scope follows the final report, Chapters 1–3 and 7.

The proof binds policy, snapshot, and scope

For a fixed policy hash, snapshot root, and capability scope:

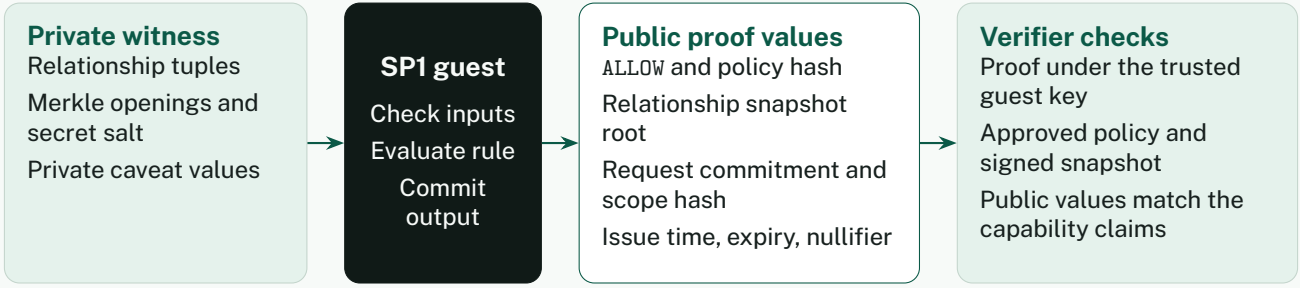
$H_P(P) = h_P$	P matches the approved canonical policy hash.
$\mathrm{Open}(w, r_G) = 1$	Private witness w belongs to the committed relationship snapshot r_G.
$\mathrm{Eval}(P, w, q) = \mathrm{ALLOW}$	The interpreter authorizes the exact bound scope q.

The verifier resolves the approved policy locally. The private relationship witness stays with the owner.

Final report, Section 4.1.

What goes into a capability proof?

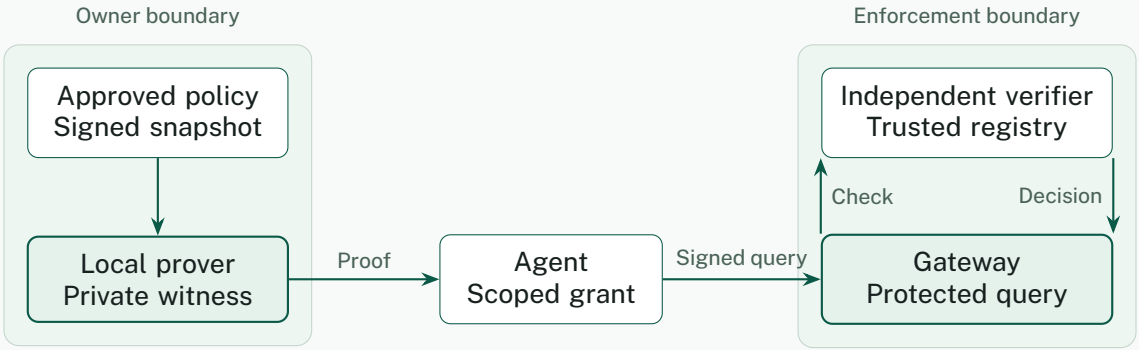
The SP1 guest is the fixed program that checks whether private relationship evidence satisfies the approved rule for the requested scope.



The guest also reads the compiled policy and issuance request. **The verifier checks the proof and public claims. The private witness stays with the owner.**

Final report, Sections 4.1 and 4.4–4.6.

Proving and enforcement have separate trust boundaries



The verifier checks the proof independently using its trusted local registry.

Final report, Section 4.2.

Where a Raziel ledger read fits in SAFR

	What SAFR asks for	What Raziel does
Governance Envelope	Submit the proposed action, its trace, and context.	The signed query and capability carry the action and authorization context: agent, task, dataset scope, and expiry.
Controls Repository	Find the rules and limits that govern the action.	The verifier resolves the capability's policy hash against its trusted registry of approved policies.
Disposition Engine	Evaluate the action against those controls before execution.	The verifier checks the proof. The gateway admits or rejects each signed read.

We prove the policy decision at issuance and enforce the grant on every read.

SAFR white paper, v1.1, pp. 9–12 · Final report, Sections 4.2 and 4.4–4.6

Capability reuse keeps request checks in place

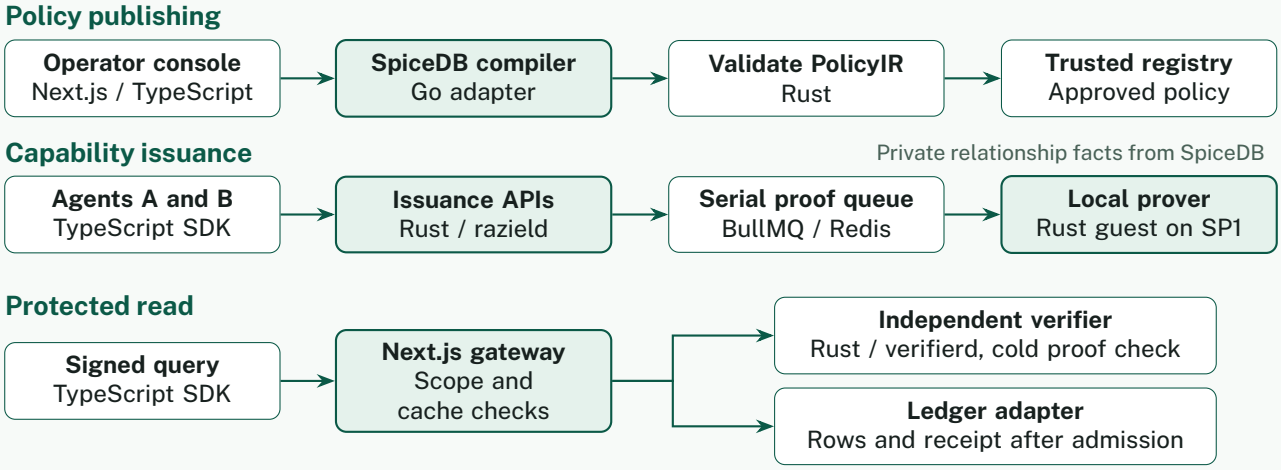


Both paths check the requested dataset binding before release.

A fresh signature authenticates each request. The grant stays bound to the original relationship snapshot.

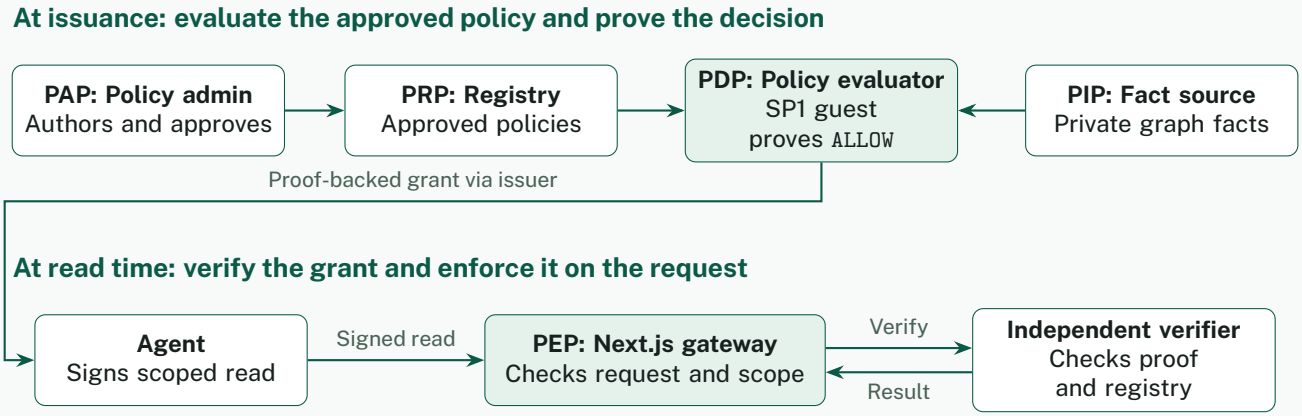
Final report, Sections 4.4 and 4.6.

Raziel's implementation at a glance



Final report, Chapter 5. Protobuf defines internal service contracts.

Policy roles in Raziel



The PEP releases rows only after the grant and signed request pass their checks.

The independent verifier checks the proof and matches its public claims to the approved policy and signed snapshot.

Experimental design

What each run tests

Guest execution

Time the authorization guest in execute-only mode.

Local PLONK proving

Measure proof generation time, peak memory, and proof artifact size.

Isolated verification

Check a retained proof offline with the issuer and prover stopped.

Semantic check: Live SpiceDB and Rust PolicyIR agreed on 21 fixture cases. A separate check compared the host evaluator with the SP1 guest.

VM profile: SP1 6.3.1, Rust 1.97.1. The 13 Aug host had 16 CPUs and 62.79 GiB RAM. The 8-CPU proof container had a 56 GiB limit and no swap.

Integration check: Synthetic ledger revisions, checksums, schema fingerprints, and row counts trace the protected reads.

Report Ch. 7 and App. A. Backend benchmark: cluster-20260813T063649Z-8cpu.

Pitfalls that shaped the implementation

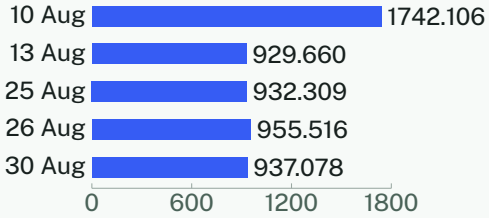
	What went wrong	What we changed
Proving memory	Early PLONK attempts ran out of memory, even at 36 GiB.	Moved proving to the NUS VM. Checked actual 56 GiB and zero-swap limits at startup.
Witness privacy	SP1 Core exposed witness-derived trace openings.	Used local PLONK for issued capabilities. Kept the witness with the owner.
Proof retries	After an uncertain reply, a new signed request queued a second proof.	Reserved capacity before signing. Persisted the exact signed attempt and looked up its job on retry.
Guest identity	A rebuilt guest no longer matched the approved verification key.	Pinned the approved guest and checked its key before sending private input.

August–October monthly logs and Second Interim Report.

Local PLONK proving results

Five individual runs, August 2026

Proof generation (s)



Run (Aug)	CPUs	Peak GiB	Artifact (B)
10 Direct	4	43.12	2537
13 Direct	8	42.46	2537
25 Retained	8	43.09	2536
26 Service	8	43.78	2536
30 Portal	8	n/r	2536

n/r = not reported.

26 Aug: Guest execution took 458 ms. Proof generation took 955.516 s.

Report Ch. 7 and App. A. Direct and retained-proof runs (10/13/25 Aug), service run (26 Aug), portal run (30 Aug).

Isolated verification and negative controls

Retained proof, 25 August 2026

447 ms isolated verification

Issuer and prover stopped. No network or swap, read-only root. The worker peaked at 7.77 MiB under a 2 GiB limit.

Isolated verifier: rejected

Eight mutations: proof, public journal, verification-key identity, policy, snapshot root, scope, expiry, and request signature.

Service and gateway: rejected

Service: altered capability and wrong signing key.

Portal: out-of-scope, replayed, stale, altered, and unsigned requests returned no transaction rows.

Proof format: SP1 Core exposed trace openings. We issued PLONK proofs to keep the relationship witness private.

Report Ch. 7. Isolated verifier (25 Aug), service run (26 Aug), portal run (30 Aug).

Capability reuse and cost interpretation

One proof supported two protected reads

Observed: A cold verified read and a cache hit used the same capability. Each read used a fresh signed request and still faced scope checks.

Model: One proof and one cold verification across N valid queries within the same grant's validity window.

$$T(N) = P + V + \sum_{i=1}^{N} (S_i + D_i)$$

P: proving. V: cold verification. S_i: request checks. D_i: data operation.

13 Aug example: $P = 929.660$ s and $V = 0.438$ s (direct check). Total setup: 930.098 s. Amortized setup: **9.30 s/query** at $N = 100$ or **0.93 s/query** at $N = 1000$.

Report Ch. 7. Direct 8-CPU run (13 Aug) and portal cache-hit record (30 Aug).

Open questions: from Raziel to Hypha

TEE-based discovery and relevance

Explore dataset metadata in a TEE. Bind query pipeline proofs to committed inputs and outputs.

Privacy Budget Sentinel

Can we track and cap what repeated queries reveal?

Federated trust registry

Can a shared ledger carry policy approvals, snapshot attestations, and revocations across firms?

ZK proof ledgers

Can partners check past proofs and receipts without relying on the issuer's live service?

Can partners verify a decision without sharing private records or transferring access?

First and Second Interim Reports. Final report, Chapter 8.

A defined workflow.
A permission another party can check.

Design partnerships and research sponsorship

raziel.gpaul.cc



Raziel



NUS National University of Singapore | FinTech Lab School of Computing

Prof. Jungpil Hahn (Advisor) Gita Alekhya Paul