From Logs to Proofs

Cryptographic Compliance for Agentic Payments

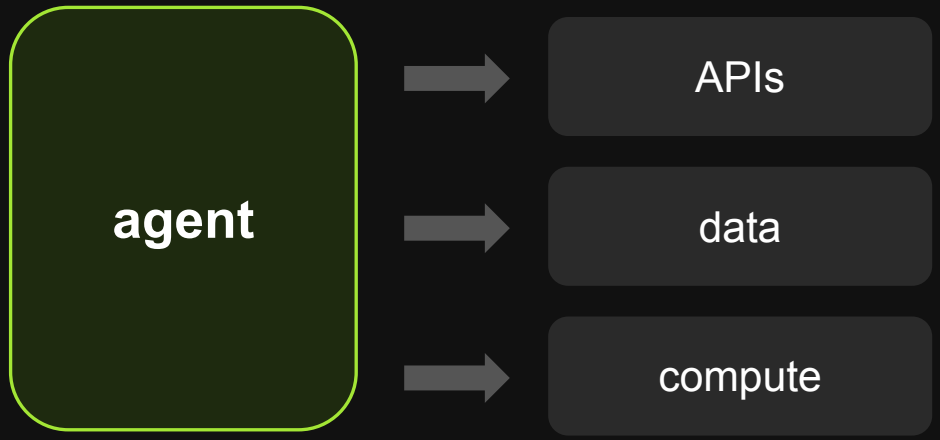
Aggre — Lemma / FRAME00, INC.

AGNTCon + MCPCon Japan 2026

a new customer

the AI agent

agents pay



under delegated authority

x402

HTTP-native payment standard

autonomously

no human in the loop

machine speed

continuous decisions · repeated payments

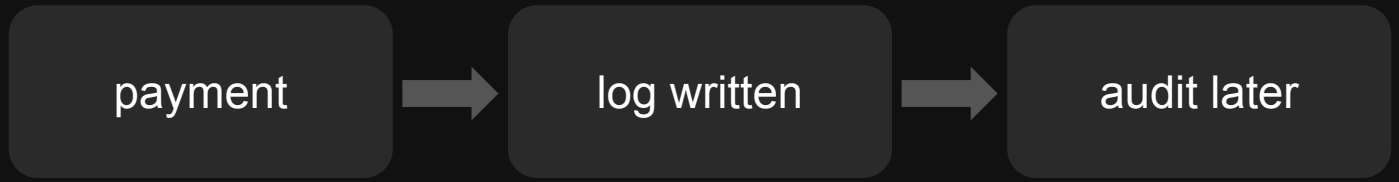
one question

who authorizes each one?

the agent chooses what to try

the owner defines what is allowed

today: logs



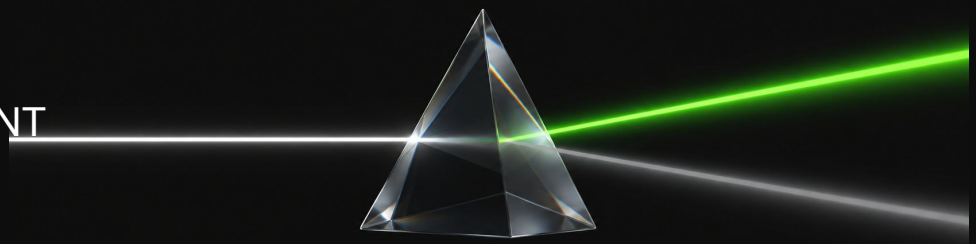
A writable log can be altered or deleted. A later audit cannot undo execution.

protect the logs · enforce the action separately

before, not after

REQUIREMENTS
purchaser role
$10 ceiling

PROPOSED PAYMENT
financial data
$0.01



PASS
sign and pay

Lemma

BLOCK
stop here

The gateway checks this payment against delegated authority.

Lemma

an execution gateway for autonomous systems

Trust402: x402 payments

```
const proofFetch = wrapFetchWithProof(
    fetch, artifact, gate, lemmaClient, proofOptions
);
return wrapFetchWithPayment(proofFetch, client);
```

Trust402 reference demo · buyer-side fetch composition

no proof, no payment

```
const proofFetch = wrapFetchWithProof(fetch, options);

const paymentFetch =
  wrapFetchWithPayment(proofFetch, wallet);

await paymentFetch(request);
// proof fails: underlying fetch never runs
```

Reference integration with fail-closed control flow

$0.01 ✓

reference demo · financial data · $10 gate

$500 → stop

reference demo · role proof fails

why a proof?

reduce trust in the party producing the evidence

verify the condition itself

A signed decision authenticates the issuer’s assertion.
A policy proof lets the verifier check the encoded condition.
Keep the authority source trusted; reduce trust in the evaluator.

Signed decision
issuer says “allow”

Policy proof
condition can be checked

correctness over confidence

soundness: false statements should not pass

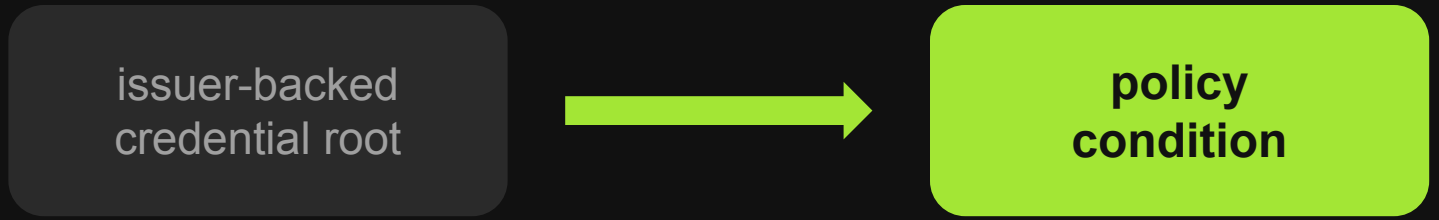
a small circuit proves the pattern

```
roleHash === requiredRoleHash;

component leq = LessEqThan(128);
leq.in[0] <== spendLimit;
leq.in[1] <== maxSpend;
leq.out === 1;
```

demo circuit · role match + one per-request spend ceiling

bind authority, not just labels



The proven fields must belong to accepted delegated authority.

A new hash chosen by the requester is not an authorization grant.

verify it yourself

```
import { groth16 } from "snarkjs";
const valid = await groth16.verify(
  verificationKey, publicSignals, proof
);
if (!valid) throw new Error("Invalid proof");
```

Standard verification API · the private witness is not an argument

enforce the verified payment

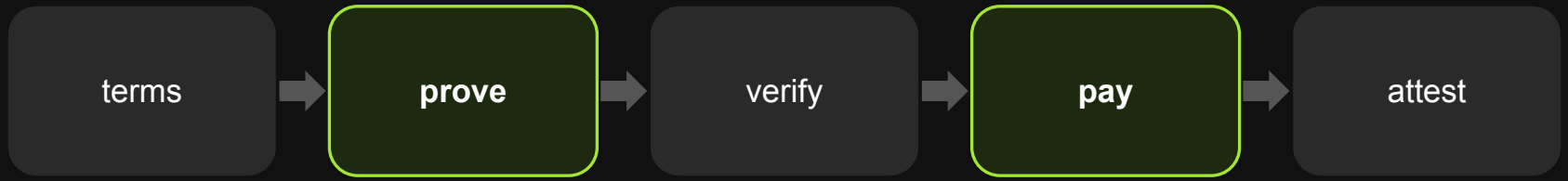
```
const terms = canonicalPayment(challenge);
const statement = statementFor(terms, delegation);
await verifyEvidence(proof, statement, trustedConfig);
await authorizeAndReserve(statement, currentState);
return signer.signAndPay(terms);
```

Target design pseudocode · same terms through verification and signing

failure must not become permission

Alter the log	policy stays unchanged
Alter the proof	verification rejects
Remove the proof	execution stops

one payment, one evidence trail



Target lifecycle · bind evidence to this payment

Short-lived authorization; retain proof, public inputs and the settlement reference.

evidence survives the boundary

partners and auditors can check the same statement

verify more. disclose less.

zero knowledge keeps private inputs out of verification

where should proving run?

Proving belongs in the latency and trust budget.
Measure generation, verification and network time separately.
Local and hosted proving expose different operational risks.

Local prover
control witness access
operate compute

vs

Hosted prover
offload compute
trust witness handling

more policy can become proof

recipient · asset · expiry · regulatory predicates

policy and proof system versions

Pin the circuit, key, and input schema

Test migrations against the same policy outcomes

A post-quantum backend requires migration and review

enterprise operation

Policy owners approve rules

Operators protect the verifier and signer

Circuit changes receive security review

Failures produce alerts and receipts

from reference code to deployed control

Trust402 makes the new control model inspectable.
Production policy can be broader than this demo.

Reference demo
role + spend ceiling
working fetch gate

Enterprise deployment
richer predicates
protected signer

test errors and adversarial inputs

```
forged evidence    => reject
changed payment    => no signature
missing evidence   => no signature
replayed request   => no second payment
concurrent spend   => no budget overrun
```

Target acceptance criteria · evaluate the protected signing path

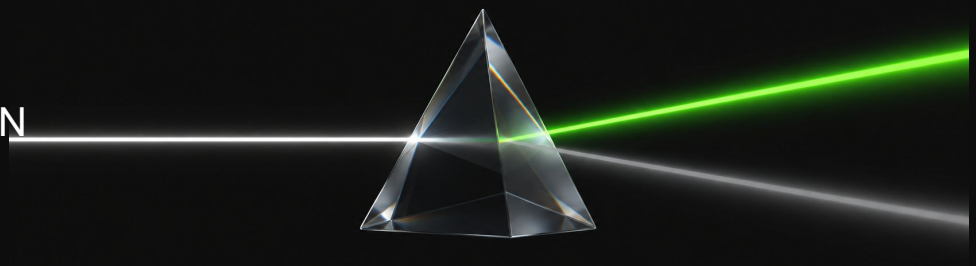
who authorizes each one?

the execution gateway checks evidence and policy

the same gate, beyond payments

REQUIREMENTS
approved source
current version

PROPOSED ACTION
use an AI answer
process a dataset



PASS
allow execution

Lemma

BLOCK
stop here

The gateway checks an action against defined, verifiable requirements.

proofs move control before execution

logs explain later · proofs decide now

run it. inspect it.

github.com/lemmaoracle/trust402

lemma.frame00.com/trust402