BUIDL CTC 2026 FALL · AI TRACK

ProofSettle

A settlement rail that pays for off-chain AI compute only when **two independent proofs, of two different kinds, agree inside a single transaction.**

Live on Creditcoin CC3 testnet | Real Attestcoin proofs | Real TEE, Google Confidential Space, AMD SEV | Verifiable in a browser

TARGET USER

Amara prices loans for people with no credit file

She runs risk at a thirty-person lender in Lagos making working capital loans to small traders. **Two thirds of her applicants have no credit history at all.**

The only way to price them is to run a model over what they do have: mobile money records, supplier invoices, till takings. She has no GPUs and no ML team, so she would have to buy that inference from someone else.

She cannot. It is her borrowers' data, she is regulated on it, and a provider's promise not to keep a copy is only a promise. And if she bought it anyway, she would be paying up front for a result she has no way to check.

So she does not. The loans do not get made.

What she needs

To name the exact code she is willing to trust, at the moment she pays, and have that enforced by something other than a supplier's word.

The other side of the rail

The provider selling her that compute has the mirror problem: today they cannot get paid without asking a stranger to trust them first.

Throughout: an **enclave** is a sealed compartment on a server that can prove which code is running inside it, and which nobody, including the machine's owner, can see into.

THE PROBLEM

Two facts have to be true, and they live on different systems

Did the payment really happen?

It happened on another chain. The contract that would release the work cannot see it, so somebody has to be trusted to say so.

Did the right code really run?

It ran off chain, in someone else's data centre. The buyer gets an answer and a bill, and no way to tell which model produced either.

Solve one and you still have the other. **Most designs solve one**, then close the gap with a trusted coordinator, which is the thing everyone was trying to remove.

Both usual fixes replace a cryptographic question with a reputational one: trust the provider, or trust an oracle operator who watched them.

HOW IT WORKS

Compose two proofs, settle on the pair

In plain terms: **the money is held by a contract, not by either party**, and it only moves when two separate pieces of mathematics agree that the buyer paid and that the agreed code is what produced the answer.

ETHEREUM SEPOLIA	OFF CHAIN	ATTESTCOIN ORACLE	CREDITCOIN
Amara pays, and names the enclave build she will accept	**A TEE runs the job and signs the result with its verdict**	**Proves the Sepolia payment to Creditcoin**	**Both proofs verified in one transaction, then money moves**
The requirement travels inside the payment event itself.	Google Confidential Space, AMD SEV. The key never leaves the enclave.	Merkle inclusion plus continuity. No oracle operator.	Accepted pays the provider, rejected refunds, partial splits.

The proven foreign event does not just carry a fact. **It carries the buyer's policy**, and the contract on Creditcoin enforces that policy against a second, unrelated cryptographic system.

Neither half settles alone.

DEPTH OF ATTESTCOIN UTILISATION

Most integrations ask the oracle for a fact

1	Call the prover, display a proof. The oracle is a widget.	SHALLOW
2	Burn on Sepolia, prove, mint on Creditcoin.	THE TUTORIAL
3	Prove a domain event and drive real application logic from it.	GOOD
4	**The proven event carries policy, enforced against a second independent proof system.**	PROOFSETTLE

The plain version: everyone else uses the oracle to **look something up**. This uses it to **carry an instruction**, and the instruction is the buyer's own.

Levels 1 to 3 all treat the oracle as a source of facts. At level 4 the contract holds no opinion of its own: it reads `requiredMeasurement` out of the proven Sepolia event and enforces exactly that. **Change what the buyer asked for and the settlement rule changes with it, with no redeploy and no admin key.**

LIVE TODAY

A real job, paid on Sepolia, settled on Creditcoin

MEASURED, NOT ASSERTED

How much of the Attestcoin oracle is real application use

142
contracts called the verifier in 17.4 days

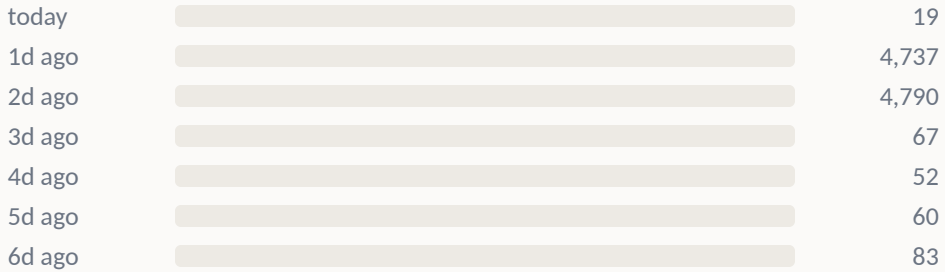
93.8%
of that traffic came from just two of them

134/142
have exactly one sending address

11
senders on the most-used contract, which is the tutorial

Before building on a protocol it is worth knowing who else is, so I measured it rather than guessing: every event the verifier precompile emitted on CC3 testnet across 100,000 blocks, complete coverage, no sampling, reproducible with `script/measure.ts`.

Almost every contract there is exercised only by the address that deployed it. That is builders testing their own work, not users using it. **Which is the headroom this was built for, and I would rather say so than imply a crowd.**



EVIDENCE

A green test suite is not evidence

49

tests, every guarantee paired with a negative case

11/11

mutations killed. Each removes one guarantee and proves the test defending it goes red

2

independent implementations of the signing digest, checked against each other

The test that matters most

Anyone can deploy their own escrow, emit a perfectly well formed payment event naming themselves as provider for any amount, and obtain a completely genuine Attestcoin proof of it. **The proof is real. The settlement would be theft.** Pinning the emitting contract is what makes a true proof also a relevant one.

Checked against the live network, not the docs

- ✓ A real proof verifies on chain: the precompile returns true
- ✓ A corrupted proof reverts with "Merkle proof validation failed"
- ✓ Sepolia is chain key 1, read from the ChainInfo precompile
- ✓ Attestation runs 7 to 9 minutes behind, measured, not quoted

THE TRUST BOUNDARY

What this does not do

No hardware attestation on chain

Checking the hardware's own certificate the way a browser checks a website's would cost far more gas than any settlement is worth. No smart contract does this today, and one that claims to is worth reading closely.

The registry records the binding, a hash of the attestation document and where to fetch it. The contract enforces the binding. **Anyone can recheck the evidence and disagree in public.**

The rail is one-directional today

Attestcoin carries attested data into Creditcoin. Writability, which would carry a settlement receipt back out, is out of scope this season, as the protocol team confirmed at the kickoff AMA.

So the return leg is a plainly labelled conventional relayer, and the buyer's refund is gated on elapsed time rather than on a receipt. **Writability is the roadmap step that removes the last trusted component.**

WHO THIS BRINGS TO CREDITCOIN

Escrow against verified delivery is a credit instrument

Creditcoin exists because credit cannot flow to people whose creditworthiness cannot be verified. ProofSettle is the same shape applied to work instead of to borrowers. **Amara is not an incidental example: the reason her borrowers cannot be scored is a compute problem, not a credit one.**

Lenders scoring thin files

Exactly Creditcoin's existing constituency, blocked on a step that happens off chain and cannot be checked.

Compute providers

Independent GPU and TEE operators who cannot prove they ran what they claim, and are paid on trust or not at all.

Agent-to-agent payments

An autonomous buyer cannot read a reputation score. It can read a signature from a named enclave build.

ROADMAP

Where this goes after the hackathon

Shipped	**Two-proof settlement live on CC3 testnet**, with a public page anyone can verify against Deep Attestcoin use that is not the tutorial
Next	**Attestcoin writability closes the loop**, replacing the conventional return leg Removes the last trusted component
Next	**Settle in an asset that already has value**, through Creditcoin's own DEX Real money in the existing economy, not a token this project invented
Then	**More source chains** as Attestcoin adds them, same contract Creditcoin becomes the settlement layer for multi-chain compute
Then	**A provider marketplace**: enclave builds compete on measurements A reason for compute operators to hold and spend CTC
The test that counts	**One real lender, one real scoring model, mainnet** Everything above is scaffolding for this

CHECK IT YOURSELF

Nothing here asks you to take my word for it

Every claim in this deck has something you can click or run. That is the point of building it this way, and it is the only reason any of it should be believed.

Click

The settlement, on Creditcoin's block explorer, with its four logs.
The payment, on Etherscan.
The refusal, a public transaction that failed for the right reason.
The page, which re-runs every check in your browser and verifies Google's attestation signature while it is there.
The repository, including the commit history that shows this happening.

Or run

```
forge test
./script/mutation-check.sh
node enclave/verify-token.mjs enclave/attestation.jwt
node site/test-attestation.mjs
npx tsx script/measure.ts
```

49 tests. 11 mutations, each removing one guarantee and proving the test defending it goes red. The attestation checked against Google's live keys. The protocol measurement reproduced from scratch.

Written by one person in nine days, and the repository is the evidence for that too rather than a claim on a slide.

IN ONE LINE

Most Attestcoin integrations ask the oracle for a fact.
This one asks it for a policy, and enforces that policy against a second, unrelated cryptographic system.

So that Amara can buy an inference from someone she has no reason to trust, and pay only when both proofs agree she got what she paid for.

Deployed on CC3 testnet | One job settled end to end | 49 tests, 11 mutations killed | Verifiable by anyone, in a browser