Tell: sealed orders that take their place in line before anyone can read them

Tell is a network the project runs itself, built from Solana's validator software (Agave). An order is sent first as a fingerprint only; the chain gives that fingerprint a place in line; after the place is final the sender opens the order; execution follows the line. Every coin on this network is test money from its faucet. This paper states only what runs and what has been measured, and how anyone can check both with the tools they already have.

What runs today

The network is four validators, all run by the project, built from Agave v3.0.9 (upstream commit 3454667767e36a915e174124c04f7e320bf37c1a) with three patch files and nothing else. The validator carries two storage changes and no change to consensus or execution: the lowest allowed ledger size is lowered (one constant, `DEFAULT_MIN_MAX_LEDGER_SHREDS`, from 50,000,000 to 100,000), and the ledger database's write-ahead log is capped at 1 GiB instead of 4 GiB, so a small network can bound its own disk. The genesis tool carries one fix: a program installed with no upgrade authority is written with no authority at all, instead of the all-zero key. The build is recorded with the hash of every patch and binary, and the stock Solana CLI 3.0 and web3.js work against the network with only the RPC address changed.

The network's final genesis has the hash Bvfk4a7a77yM1p6p9ZqosvqCiddsQh5GaPi5CXQY2ybY and its slot 0 is dated 2026-11-25T01:07:37Z (UTC, the block time); it is never reset, so every order since then stays countable.

Four programs are part of the network's genesis, installed under the upgradeable loader with no upgrade authority, so nobody, the project included, can change them:

- the envelope program: `seal` takes a fingerprint, a public cap ("this order is not larger than X") and a bond, and gives the next place in that market's line; `open` takes the order and its secret within the open window and checks them against the fingerprint; `execute`, which anyone may call, runs opened orders strictly in place order, returns their bonds, and burns the bond of every envelope that was not opened;
- the sealed pool: a constant-product pool with a 0.25 % fee whose swap, add and remove entries accept only the envelope program's signer, so an order that wants to step in front must itself be sealed, blind and with a bond: sealed after the order it targets, it gets a later place; sealed ahead of it, it must guess, and every side it does not open loses its bond;
- the open pool: the same pool with public entries, kept beside it for comparison;

- the sealed auction: lots whose bids are envelopes, opened only after the lot closes (behind a switch, not open yet).

The fingerprint is `sha256("blindfold:v1" || line || owner || length || order || secret)`, where the secret is 32 random bytes made in the sender's browser. Until the sender opens it, the content of an order is known only to the sender: not to the validators and not to the project. The test assets are TEST (the faucet's money) and ORE (the traded asset); amounts are always shown with their unit.

Measured

Every figure below is a rig record: a command that produces the number on its own, run several times, saved with the machine and the conditions. Every row comes from the final network (genesis Bvfk4a7a..., slot 0 at 2026-11-25T01:07:37Z). The traffic is the project's own labelled bots: a liquidity bot, a target bot that sends the same large orders to both pools, and a sandwich bot that tries to cut in on both. None is copied from elsewhere. Amounts are faucet test money (TEST).

The compared pair is SLAB/TEST, live since slot 9,390 (2026-11-25T02:12:15Z), whose sealed pool has an open window of 12 slots; the bots trade it from 2026-11-25T02:13:43Z. The first pool, ORE/TEST, has an open window of 4 slots; its figures from 01:10Z to 01:53Z are given after the table.

Record	Value	Conditions
`line-order`	0 orders out of line (p50 and p99 of 5 runs; claim 0, backed)	every sealed-pool order the node held at 2026-11-25T02:52Z, slot 0 to slot 15,214: 145 executed and 90 skipped (not opened, bond burned) on three lines - SLAB/TEST 67 and 21, ORE/TEST 77 and 67, and the project's own test holder pool 1 and 2 - recounted from the blocks
`seal-to-execute`	7042.5 ms (p50 of 10 runs; p99 8627 ms)	SLAB/TEST, one small sealed buy by the project's target bot through the site's relay, from sending the seal to the executed state; the 12-slot open window, the open and the execution included
`sandwich-open-pool`	29515.08 TEST taken by the sandwich bot	SLAB/TEST open pool, 2026-11-25T02:13:46Z to 03:14:03Z: 61 attempts, 59 of them cut in ahead of the target order; every target order was attempted
`sandwich-sealed-pool`	-48668.07 TEST taken by the sandwich bot (a loss)	SLAB/TEST sealed pool, 2026-11-25T02:13:47Z to 03:12:11Z, every attempt of the bot: 17 attempts (11 sealed ahead of the target, 6 sealed after seeing it), placed around the target 4 times, profitable 0 times; net -48,668.07 TEST after 48,297.46 TEST of bonds burned; sampled: one target seal in 5 gets an attempt
`unopened-attack`	-36357.47 TEST profit (the attack loses)	a separate series: only the 11 attempts sealed in both directions ahead of the target (2026-11-25T02:21:14Z to 03:08:10Z), the favourable side opened after the target opened and the rest left unopened; placed around the target 4 times, profitable 0; 35,986.85 TEST of bonds burned; sampled one target seal in 5. The 6 attempts sealed after the target are counted only in `sandwich-sealed-pool`

Record	Value	Conditions
ledger-growth-per-day	-10.52 GB per day (p50 of 3 ten-minute windows; p99 53.44)	from 2026-11-25T01:53Z, ledgers at their size cap and the write-ahead log capped at 1 GiB: the whole network's disk stays between 8.5 and 9.9 GB (samples every 5 minutes, 01:55Z to 02:20Z); a ten-minute window moves up or down as the database compacts, so the figure is near zero, not a trend

On the first pool, ORE/TEST (open window 4 slots, 2026-11-25T01:10Z to 01:53Z, the same commands), the same records read: seal to execute 4183.5 ms (p50 of 10 runs); the sandwich bot took 17,115.83 TEST from the open pool in 40 attempts, 34 of them cut in; in the sealed pool it made 23 sampled attempts, was placed around its target in none, and lost 92,235.85 TEST in bonds.

How to reproduce

Every figure has the command that made it. Each prints its value on its last line as `value=<number>`, and `rig.py run --name <record> --runs <k> -- <command>` repeats it and saves p50, p90 and p99 with the host:

- `line-order` (with the claim 0): `rig.py run --name line-order --runs 5 --claim 0 -- node chain/rig/line-order.mjs --rpc https://rpc.tell.fun`. It reads blocks with plain JSON-RPC (`getBlocks`, `getBlock`), takes every `Executed`, `Skipped` and `Voided` event of the envelope program in chain order, and counts how many places in each sealed-pool line ran out of order.
- `seal-to-execute`: `rig.py run --name seal-to-execute --runs 10 -- bash worker/bots/measure/rig.sh seal-to-execute` (sends one small sealed order and times it to the executed state).
- `sandwich-open-pool`, `sandwich-sealed-pool`: `rig.py run --name <record> --runs 3 -- bash worker/bots/measure/rig.sh <record>` (sums the sandwich bot's attempts since the tally start, with each attempt's bonds read from its envelope accounts on chain).
- `unopened-attack`: `rig.py run --name unopened-attack --runs 3 -- node chain/rig/unopened-attack.mjs` (the same attempt records, only the attempts sealed in both directions ahead of the target).
- `ledger-growth-per-day`: `rig.py run --name ledger-growth-per-day --runs 3 -- bash chain/rig/ledger-growth.sh` (the size of the network's ledgers before and after a window, scaled to a day).

With the stock Solana CLI and the public RPC address, https://rpc.tell.fun (also shown on the site's Verify page; its WebSocket is wss://rpc.tell.fun):

01 `solana -u https://rpc.tell.fun program show <program id>` prints `Authority: none` for each of the four programs.

02 `solana -u https://rpc.tell.fun block <slot>` and `solana -u https://rpc.tell.fun confirm -v <signature>` show an execute transaction and its program log.

03 `node chain/rig/line-order.mjs --rpc https://rpc.tell.fun` recounts "N executed, 0 out of line" from the blocks themselves.

The source of the patches, the programs and these scripts is published with the project's public code.

Sources

- Upstream engine: Agave v3.0.9 (Anza), https://github.com/anza-xyz/agave, the release this network is built from.
- Token programs in genesis: SPL Token 3.5.0, Token-2022 8.0.0, Associated Token Account 1.1.1 and Memo 1.0.0 / 3.0.0, the program files that ship inside that Agave release.
- A different method, for comparison: Jito's Block Assembly Marketplace orders transactions inside a trusted execution environment (AMD SEV-SNP) on its own nodes, as described at https://www.helius.dev/blog/block-assembly-marketplace-bam . Tell uses no key committee and no secure chip: the sender seals and the sender opens, and in exchange the sender acts twice (the browser wallet does both steps).

What does not work yet

- The unopened-envelope hole. A party can seal several envelopes in both directions around someone else's order and, after seeing that order open, open only the favourable ones and leave the rest. The defence today is the bond: every unopened envelope loses its bond, sized to its public cap. The project runs this attack with its own labelled bot. On the main pair SLAB/TEST, whose open window is 12 slots, the bot got its pair sealed ahead of the target placed around a sealed target 4 times in 11 such attempts by 2026-11-25T03:08Z and opened only its favourable side after seeing the target open; it was profitable 0 times, and in total it lost 36,357.47 TEST once the bonds of the unopened sides burned (the record `unopened-attack` above, sampled one target seal in 5). The wider the window, the more often this placement succeeds. On the first pool, ORE/TEST, with a 4-slot window, it was never placed around its target. That is a measurement of one bot at one bond rate, not a proof: the honest sentence is "to cut in you must commit blind and post a bond", not that cutting in is ruled out.
- The sender must open. An envelope is opened by the browser that sealed it, inside a short window of slots. A closed browser cannot open, and the bond of an unopened envelope is burned; the wallet warns before the page closes.
- What the claim covers. It holds for orders of programs that accept envelopes: the sealed pool, and the sealed auction behind its switch. Other transactions on the network are ordinary transactions.
- Who runs it. All four validators are the project's. Outsiders reach the network only as RPC clients; nobody else validates it yet.
- What it is not. The seal is a fingerprint and an opening, not encryption. Balances are not hidden. It is not Solana mainnet and not a service on it; it is a separate network built from the same software. Every coin on it is faucet test money with no price, so nothing here is a figure in any currency.

- History. A node keeps only hours of history; the project keeps its own index from genesis, and the recount above reads what the node still holds.
- Not measured yet: the slot time as a rig record, and the network under load from many outside users. Their rows are added when their runs finish.