Sealed by default: a Solana-fork chain where every balance of its main asset is ciphertext

A chain run as a fork of the Solana validator client whose main asset moves by confidential transfers on every path the project ships: its amounts and balances exist on chain as ciphertext, every transfer carries zero-knowledge proofs that anyone can check, and only the holder of an account can read its balance. The native coin that pays fees is public. This paper states what runs today, what was measured on the running network, how to repeat each figure with tools people already have, and what does not work yet.

What runs today

The engine. A fork of Agave, release `v4.5.0-alpha.4` (commit `ccddd3dffc7edb1d6ed8182ac8e9cbd73b24b159`), with one patch that lowers the validator's minimum ledger-size floor so a small network can keep a bounded ledger (`ledger-floor.patch`: `LEGACY_DEFAULT_MIN_MAX_LEDGER_SHREDS` 50,000,000 -> 100,000 shreds). Nothing in the transaction path is changed: confidential transfers are Token-2022's own, and their proofs are checked by the engine's native ZK ElGamal Proof program.

Genesis. Token-2022 v8.0.0 (the published release build), the Associated Token Account program v1.1.2, the SPL Record program (used by the current `spl-token` CLI to upload the range proof), and the engine's own core programs (stake, address lookup table, config, feature gate). The legacy SPL Token program is not in genesis: there is no public token program on this chain. Transaction format v1 (up to 4,096 bytes) is active from the first slot.

The main asset. One Token-2022 mint, `TEST` (a test coin, 6 decimals), with the confidential-transfer extension on, accounts approved automatically and no auditor key. Its whole supply, 1,000,000,000 TEST, was minted once into the faucet's sealed reserve by a recorded setup script in the first slots; the mint authority was then removed, so the supply is fixed and anyone can read that it is: mint authority, freeze authority and confidential-transfer authority are all unset. Every account the site, the faucet and the wallet create is configured for confidential transfers and refuses public credits.

The native coin. `GAS` (test gas) pays fees and is public. It is never presented as hidden. Neither `GAS` nor `TEST` has value; the chain runs on test coins.

What a sealed transfer is. The sender's software builds three zero-knowledge proofs (equality of the new balance's ciphertext and commitment, validity of the amount's ciphertexts, and a range proof), places them in proof accounts and sends the transfer, which references them. The official `spl-token` CLI does this in 10 transactions; the site's in-browser wallet in 4. The amount lands in the receiver's pending balance as ciphertext; the receiver applies it to

his available balance. Only the owner's keys open the balance (ElGamal for pending amounts, AES for the available balance), in his own software.

What everyone sees. Both token accounts and their owners, the mint, the slot and time, the fee payer and its public fee, that the transaction was a confidential transfer, and the receiver's incoming-transfer counter. Amounts and balances appear only as ciphertext. Who sent to whom is visible.

The network. 4 validators, all run by the project on one server, inside a private network; one bootstrap validator in genesis and three joined and staked with the stock CLI, with equal stake once warmed up. The public reaches the chain only through one RPC endpoint (JSON-RPC and WebSocket) behind a gateway that refuses node-address and admin methods and writes no per-request log. The RPC node serves under an hour of transaction history (see *Measured*), not the full history.

Checked with the stock tools. A check run against the public RPC URL with the stock `solana` CLI and `@solana/web3.js`, passed for node health (4 validators, none delinquent), a CLI transfer, a web3.js transfer, and a confidential transfer read back with `solana confirm -v`.

Measured

Every figure below was measured on the running 4-validator network with `rig.py`, which records each run with the host it ran on: AMD Ryzen 9 9950X (28 vCPU, AVX-512), 62 GB RAM, one host. Times are wall time; p50 and p99 over the runs shown.

Figure	Value	Conditions
`sealed-transfer-txs`	10 transactions	one confidential transfer of 1 TEST with the official `spl-token` CLI 5.7.1; 5 runs
`sealed-transfer-bytes`	5453 bytes in total	the same transfer, all 10 transactions as sent; 5 runs
`sealed-transfer-cu`	268077 compute units in total	the same transfer, read from each transaction's metadata; 5 runs
`sealed-transfer-fee`	70000 lamports of GAS	the same transfer (14 times a public transfer's fee); 5 runs
`sealed-transfer-secs`	3.9245 s p50, 4.217 s p99	the CLI's wall time for the whole transfer, each step confirmed; 10 runs
`sdk-sealed-transfer-txs`	4 transactions	one confidential transfer of 1 TEST by the site's in-browser wallet client (proofs in context-state accounts); 5 runs

Figure	Value	Conditions
sdk-sealed-transfer-bytes	3529 bytes in total	the same wallet transfer, all 4 transactions as sent; 5 runs
sdk-sealed-transfer-cu	263798 compute units in total	the same wallet transfer; 5 runs
sdk-sealed-transfer-fee	35000 lamports of GAS	the same wallet transfer (7 times a public transfer's fee); 5 runs
public-transfer-txs	1 transaction	one public transfer of GAS with the stock `solana` CLI 3.0.0, for comparison; 5 runs
public-transfer-bytes	215 bytes	the same public transfer; 5 runs
public-transfer-cu	150 compute units	the same public transfer; 5 runs
public-transfer-fee	5000 lamports of GAS	the same public transfer; 5 runs
public-transfer-secs	0.6345 s p50, 3.714 s p99	the CLI's wall time for the public transfer; 10 runs
wasm-node-proof-ms	42.955 ms p50, 44.04 ms p99	generating the three proofs of one transfer with the official WASM build `@solana/zk-sdk` 0.5.4 in Node, one thread; 10 runs
sdk-browser-transfer-proof-ms	51 ms p50, 62 ms p99	the same three proofs generated in headless Chromium 158 with the same WASM; 5 runs
sdk-browser-sealed-transfer-proof-ms	62 ms p50, 69 ms p99	proof generation inside a full in-browser sealed transfer (4 transactions); 3 runs
sdk-browser-sealed-transfer-total-ms	1833 ms p50, 1881 ms p99	the in-browser wallet's whole sealed transfer: proofs plus 4 transactions, each confirmed; 3 runs

Figure	Value	Conditions
`rpc-history-window-min`	48.3 min	how far back the RPC node serves transactions (current slot minus the first available block, at the measured slot rate of about 5.2 slots per second); bounded by the node's ledger size limit; 1 run
`gateway-75-clients-read-p99-ms`	154.5 ms	p99 latency of page reads (slot, performance samples, an account, its signatures) by 75 concurrent clients in a closed loop for 30 s, measured at the gateway inside the server, not through the public edge; 1 run

Browser figures were measured in a headless browser on the same server, not on a phone or a laptop.

How to reproduce

Every command below uses only public tools. `$RPC` is the public RPC URL shown on the site's *Check it yourself* page. The `TEST` mint is `5n31rh6t9mgFYZRhoSb8k2xUcdCgmDL5BFZbjvkPLHif`; the Token-2022 program is `TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb`. Use `spl-token-cli` 5.7.1 or later (older releases build proofs in a format this engine rejects; see *Sources*).

Get test gas and seal an account:

```
solana-keygen new -o me.json
solana --url $RPC airdrop 1 me.json
spl-token --url $RPC --program-id
TokenzQdBNbLqP5VEhdkAS6EPFLC1PHnBqCXEpPxuEb create-account --owner
me.json --fee-payer me.json 5n31rh6t9mgFYZRhoSb8k2xUcdCgmDL5BFZbjvkPLHif
spl-token --url $RPC configure-confidential-transfer-account --owner
me.json --fee-payer me.json 5n31rh6t9mgFYZRhoSb8k2xUcdCgmDL5BFZbjvkPLHif
spl-token --url $RPC disable-non-confidential-credits --owner me.json --
fee-payer me.json 5n31rh6t9mgFYZRhoSb8k2xUcdCgmDL5BFZbjvkPLHif
```

Test coins come from the site's faucet (a sealed transfer into the account's pending balance), then:

```
spl-token --url $RPC apply-pending-balance --owner me.json --fee-payer
me.json 5n31rh6t9mgFYZRhoSb8k2xUcdCgmDL5BFZbjvkPLHif
spl-token --url $RPC display <your token account>      # Balance: 0, the
rest only as ciphertext
```

A public transfer into a sealed account is refused (Token-2022 `custom program error: 0x31`): move one coin of your own to your public balance (that amount becomes public) and try it:

```
spl-token --url $RPC withdraw-confidential-tokens --owner me.json --fee-payer me.json 5n31rh6t9mgFYZRhoSb8k2xUcdCgmDL5BFZbjvkPLHif 1
spl-token --url $RPC transfer --owner me.json --fee-payer me.json 5n31rh6t9mgFYZRhoSb8k2xUcdCgmDL5BFZbjvkPLHif 1 <another sealed owner>
```

A sealed transfer, and how to read it as anyone else does:

```
spl-token --url $RPC transfer --owner me.json --fee-payer me.json 5n31rh6t9mgFYZRhoSb8k2xUcdCgmDL5BFZbjvkPLHif 1 <another sealed owner> --confidential
solana --url $RPC confirm -v <the "confidential transfer" signature it prints>
```

The figures in *Measured* come from these runs (`rig.py run --name <figure> --runs <n> -- <command>`):

- `sealed-transfer-*`: one `spl-token transfer ... --confidential` per run; transaction count, bytes, compute units and fees read back with `getTransaction` for every signature the CLI printed; seconds = the CLI's wall time.
- `public-transfer-*`: one `solana transfer` of GAS per run, read back the same way.
- `wasm-node-proof-ms`: the three proofs of a transfer built and verified with `@solana/zk-sdk` 0.5.4 in Node (`zk_proof_bench.cjs`, in the public repository).
- `sdk-browser-*`: the site's own client package running the same transfer in headless Chromium.
- `gateway-75-clients-read-p99-ms`: `load-read.mjs` with 75 clients for 30 s.

The fork's patch, the genesis program list, the client package and these measurement scripts are published in the project's public repository.

Sources

- Agave `v4.5.0-alpha.4`, Anza, https://github.com/anza-xyz/agave (pre-release; the fork's upstream). Its proof verifier is the crate `solana-zk-sdk` 8.1.0.
- Token-2022 confidential transfer extension, program release `program@v8.0.0` (2025-03-26), https://github.com/solana-program/token-2022; no security advisory is published for the repository (read 2026-11-19).
- `solana-zk-sdk` 5.0.0 release notes (2026-01-13), https://github.com/solana-program/zk-elgamal-proof: "Add unique domain separator for zk elgamal proof program for proofs" and hashing the proof context into the transcript. Proofs made by earlier releases do not verify on engines built with 5.0 or later, and the reverse; `spl-token-cli` 5.3.0 (built on 2.2.1) is refused by this chain, 5.7.1 (built on 7.0.1) works.

- SIMD-0385, transaction format v1, https://github.com/solana-foundation/solana-improvement-documents/blob/main/proposals/0385-transaction-v1.md (active on this chain from genesis).
- Bünz, Agrawal, Zamani, Boneh, "Zether: Towards Privacy in a Smart Contract World", IACR ePrint 2019/191, https://eprint.iacr.org/2019/191 (2019): "We design a new smart contract that keeps the account balances encrypted and exposes methods to deposit, transfer and withdraw funds to/from accounts through cryptographic proofs."
- Solana, "Post Mortem: ZK ElGamal Proof Program Bug", 2025-06-25, https://solana.com/news/post-mortem-june-25-2025.
- Token-2022 issue 657, "Re-enabling the ZK ElGamal Proof Program and Token-22 Confidential Transfer Features", https://github.com/solana-program/token-2022/issues/657 (closed 2026-09-03).

What does not work yet

- **Who sent to whom is visible.** Both accounts, both owners, the time and the fee payer of every transfer are public. Only amounts and balances are sealed.
- **Moving between public and sealed balance is public.** Minting, a deposit into a sealed balance and a withdrawal from it show their amounts. The total supply is public.
- **A fresh account's first balance can be guessed.** The faucet's grant has a fixed size.
- **Only this asset is sealed by default.** Token-2022 can still create mints without confidential transfer on this chain; "by default" means the main asset and every path the project ships.
- **Wallets that do not know the extension** show a balance of 0 and cannot send it.
- **The proof system carries a known history of risk.** Solana disabled the ZK ElGamal Proof program after a bug was found in it in June 2025 (Solana's post-mortem of 2025-06-25) and re-enabled it later; the engine carries both steps as feature gates (`disable_zk_elgamal_proof_program`, `reenable_zk_elgamal_proof_program`), and Token-2022 issue 657 tracks the re-enabling. This chain relies on the same program and inherits any flaw still in it.
- **One operator.** All 4 validators are run by the project on one server. Distance between them is not real, and the network is not run by independent parties.
- **Under an hour of history.** The RPC node serves about the last 48 minutes of transactions, not the full ledger; older transactions are not served, by us or by anyone, unless someone indexed them while they were available.
- **The engine is a pre-release.** `v4.5.0-alpha.4` is an alpha of Agave.
- **Test coins only.** `GAS` and `TEST` have no value, and nothing on the chain is backed by anything.
- **Browser figures** were measured in a headless browser on the server, not on phones or laptops.

- **One public endpoint.** The network is reached only through one RPC URL on one server; its capacity was measured once with 75 clients reading, and a larger audience has not been measured.

