HashKinetics: A Post-Quantum, Shielded-by-Default Settlement Layer for the AI-Agent Economy

Whitepaper v1.0

Yaduvendra Mukherjee
HashKinetics · Token: $HKN

2026-08-18

Abstract

Autonomous software agents have become economic actors: they hold budgets, purchase services from one another at machine speed, and fail in ways human-oriented payment infrastructure was never designed to contain. HashKinetics is a sovereign Layer-1 blockchain purpose-built for this population. It combines three properties no other production network offers simultaneously: **post-quantum settlement** (every signature that moves money or secures consensus is hash-based; no elliptic curves, no lattice signatures), **shielded-by-default balances** (agent budgets, counterparties, and payment flows are hidden behind hash commitments and verified by STARK proofs, with scoped, holder-initiated, offline-verifiable lawful disclosure), and **consensus-enforced spending hierarchies** (organizations delegate budgets to agents through MandateTrees whose caps are enforced by validators themselves, over balances the chain cannot see). Payments meter through hash-chain micropayment channels at effectively unbounded rates and settle on-chain in single transactions. The proof system is deliberately pairing-free: the deployable artifact is always a raw STARK, and per-block recursive aggregation gives every validator exactly one constant-size proof to verify regardless of transaction count. This paper specifies the full protocol as deployed on mainnet, its cryptographic doctrine and threat model, its economics and governance, the measured performance envelope it was built against, and the phased, demo-gated engineering history that produced it.

Contents

0 Status and provenance **1**

1 Introduction: two clocks **2**

2 Design doctrine **2**

3 System overview **3**

4 Cryptographic foundations **3**
4.1 The key hierarchy 3
4.2 Domain separation 4
4.3 Ed25519, demoted 4

5 Consensus **4**
5.1 Engine and votes . . . 4
5.2 State binding . . . 4
5.3 SCMS: rotation under a stateless root . . . 4
5.4 Signature bandwidth . . . 4

6 The transparent skeleton **5**
6.1 Accounts . . . 5
6.2 MandateTree v2 . . . 5
6.3 QHT channels . . . 5

7 The shielded pool **5**
7.1 Notes, tree, anchors, nullifiers . . . 5
7.2 Transactions and the binding rule . . . 6
7.3 Addresses and spend trees . . . 6
7.4 The spend statement (circuit v3) . . . 6

8 The proving system **6**
8.1 Stack, measured . . . 6
8.2 Aggregation: one proof per block . . . 7
8.3 Verifying-key pins and version pools . . . 7
8.4 The in-circuit hash (D1) . . . 7

9 Confidentiality and discovery **7**

10 Mandates × shielded **7**

11 CVA: lawful disclosure without master keys **8**

12 Node, wire, and engineering culture **8**
12.1 The node . . . 8
12.2 The wire . . . 8
12.3 Demo-gated engineering . . . 9

13 Performance and capacity **9**
13.1 Measured envelope . . . 9
13.2 The capacity model, tiered . . . 9
13.3 The throughput program (“the 183k plan”) . . . 9

14 Economics and governance **9**
14.1 Money on the chain . . . 9
14.2 Parameters vs constitution . . . 10

15 Security analysis **10**
15.1 Threat model against a quantum adversary . . . 10
15.2 Protocol-level defenses . . . 10
15.3 Assurance . . . 10

16 The road here: phases and gates **11**

17 Related work **11**

18 Conclusion **11**

References **12**

0 Status and provenance

This whitepaper describes the complete HashKinetics protocol in the present tense of a running mainnet. In keeping with the project's documentation doctrine — dated claims, measured numbers, no promises dressed as facts — readers should note the provenance of every quantitative statement:

- **Measured** numbers (proof latencies, block cadence, aggregate sizes, test counts, demo transcripts) originate from the 2026 development network on documented hardware (AMD Threadripper PRO 7995WX, NVIDIA RTX 5090, WSL2), recorded in the repository's engineering logs (`docs/MASTER-BUILD-PLAN.md`, `zkvm-bakeoff/RESULTS.md`). They are facts about that environment.
- **Configured** numbers (block capacity, timeout pacing, tree depths, window sizes) are protocol constants chosen deliberately and changeable by documented upgrade paths.
- **Design targets** (mainnet-scale throughput, validator-set size, market behavior of fee and bond mechanisms) are engineering projections with named assumptions, validated progressively through the gate process of §16.

Nothing in this document is an offer of tokens or securities. The honesty ledger that accompanied every development release remains a maintained artifact in the repository.

1 Introduction: two clocks

Two independent developments define the moment this network was built for.

The first clock: machine customers arrived. Payment-protocol consortia for agent commerce (x402 and its foundation, with premier members spanning card networks, processors, and cloud providers; Google's AP2; the ERC-8004 identity registry) moved agent payments from thesis to industry. Cumulative agent-initiated transactions crossed the hundreds of millions. The failure data arrived on schedule: a majority of enterprises deploying agents report at least one agent-caused incident, and roughly a third report direct financial loss. Every widely deployed mitigation shares one architectural weakness — spending caps live in the application layer, in dashboards, SDKs, and policy engines that a compromised or jailbroken agent simply ignores. An agent holding raw key material *is* its principal; nothing between it and the ledger says otherwise.

The second clock: the quantum deadline became official. Regulators and standards bodies stopped treating post-quantum migration as optional. The IETF drafts post-quantum requirements for x402 payment *receipts*; the EU's AMLR Article 79 regime (applicable 2027-07-01) reaches crypto-asset service providers and explicitly contemplates "optional privacy features"; Ethereum's long-horizon roadmap converges on hash-based signatures and STARK proofs — the exact stack this network launched with. Meanwhile harvest-now-decrypt-later makes *confidentiality* migration urgent before signature forgery is even feasible.

At the intersection sits an unoccupied position: post-quantum **settlement** (not just receipts); privacy that is **native and default** (not bolted on, and not the un-listable absolutism that gets networks delisted); spending authority that is **a consensus rule** (not an app-layer suggestion). Each pairwise combination is structurally hard. HashKinetics is the demonstration that all three compose. The design brief, fixed at inception: *allowances for AI agents, enforced at the protocol level — confidential like a bank account, disclosable like one too.*

2 Design doctrine

Six invariants govern every design decision; they are constitutional in the governance sense of §14.

D-1 Pure hash-based signatures. Every signature that authorizes value movement or secures consensus rests on hash-function security alone: SLH-DSA-SHAKE-192s (FIPS 205) stateless roots; LMS/HSS (RFC 8554, SHAKE-256) consensus votes; WOTS one-time signatures for in-circuit spend authorization; PayWord chains for channels. **No lattice signatures anywhere** — not as fallback, not hybrid.

D-2 The doctrine split. The single non-hash primitive is ML-KEM-768 (FIPS 203), used *only* for note encryption and stealth addressing. If lattices fall, an adversary reads metadata; it never moves funds. Privacy degrades gracefully; custody does not degrade at all.

D-3 F1: no pairing wraps, ever. Popular "small proof" paths (Groth16/PLONK over BN254) are pairing-based, hence not post-quantum. The deployable artifact is always a **raw compressed STARK**; size and verification targets are met by recursive aggregation (§8).

D-4 Shielded by default. Privacy is the product, not a mode. Development networks ran transparent-first while circuits hardened — and were never marketed otherwise.

D-5 CVA disclosure. Scoped, holder-initiated, offline-verifiable; granularity of one payment or one epoch; **no master viewing key exists structurally** (§11).

D-6 Consensus-guarded stateful signatures. Leaf index = account nonce as a consensus rule; observed reuse is slashable evidence; reserve-then-sign persistence means a crash can never replay a one-time leaf.

Two hash functions serve the whole system: domain-separated **SHAKE-256 everywhere outside circuits**, and **SHA-256 inside circuits** for pool v1 — chosen by measurement (§8.4), rotatable through version pools.

3 System overview

The chain is two planes with a value-conserving bridge.

```
      TRANSPARENT SKELETON  (the machine plane)
validators & staking . MandateTree skeletons . channel anchors
CASP envelope registry . account nonces & key commitments
                  |  mint / burn  (bridge)
                  v
      SHIELDED POOL  (the user plane)
notes (hash commitments) - append-only commitment tree - nullifier set
   every spend: a STARK proof, verified by every validator
   every block: ONE recursive aggregate proof covers them all
```

The transparent skeleton carries what must be public for the machine economy to function — validator sets, mandate envelopes and their public caps, channel anchors, compliance registrations — and deliberately no user balances. The shielded pool carries value as *notes*: hash commitments binding amount, owner tag, and blinding randomness. Notes enter by minting (the mint proof is the inflation guard), move by shielded spends (nullifier in, two commitments out), and exit by unshielding (a spend whose public `fee` credits a transparent account). A fully shielded transfer ($fee = 0$) leaves **no transparent residue**: one nullifier, two commitments, nothing else.

Three actor roles surround the ledger. **Agents** prove their own spends locally (∼1.2 s on workstation GPUs, measured) and meter commerce through channels. **Validators** run BFT with hash-based votes and verify STARKs in-node; they never see balances. **Aggregators** (proposer in v1; a bonded fee-earning role at scale) fold a block's proofs into one recursive aggregate. Facilitators, view services, and watchtowers are permissionless service roles.

4 Cryptographic foundations

4.1 The key hierarchy

Every long-lived identity is anchored by a **stateless SLH-DSA-SHAKE-192s root** (48-byte public key, category 3) that never exhausts and signs only to certify. Beneath each root live stateful operational trees: validators sign every vote with LMS/HSS (∼ms sign/verify; tens of thousands of signatures per tree; rotation under the root per §5.3 — *there is no root exhaustion*); agent accounts use hash-based keys whose **leaf index is the account nonce** by consensus rule; shielded spend authority uses per-address WOTS one-time keys in spend trees (§7.3); channels use PayWord chains. One 32-byte master seed derives a validator's entire identity family; agent wallets similarly derive spend trees, nullifier keys, and per-epoch KEM keys.

4.2 Domain separation

Every hash application is domain-separated, $H(tag \,\|\, data)$, with tags drawn from a versioned registry (`hk/v1/...`) covering transaction digests, identifiers, block values, tree nodes, nullifiers, address tags, key derivations, note encryption, aggregation coverage, and verifying-key pins. Distinct uses cannot collide by construction; the registry is part of the audit surface.

4.3 Ed25519, demoted

Ed25519 exists in exactly one place: libp2p transport identity. It signs no consensus message and secures no ledger state; a discrete-log break yields endpoint impersonation — detectable and harmless to funds.

5 Consensus

5.1 Engine and votes

HashKinetics runs Malachite, a Rust BFT engine of Tendermint lineage, instantiated with a custom signing scheme: **every prevote, precommit, and proposal is LMS/HSS-signed**. Measured development cadence: ∼1.4 s blocks at round 0 with four validators, dominated by deliberately conservative pacing rather than cryptography. Consensus sign-bytes are manual, domain-tagged layouts — independent of any wire codec by construction (§12.2).

5.2 State binding

The proposer binds its parent application hash into the block batch, and batch bytes are hashed into the value identifier voted on. A diverged validator computes a different commitment and cannot vote for the proposal: **state divergence is consensus-fatal**, surfaced immediately.

5.3 SCMS: rotation under a stateless root

The Stateful Consensus-key Management System: (1) each validator's genesis entry carries its permanent SLH-DSA root beside its initial operational key; (2) rotation is a root-signed

`RotationCert` $\{root\_pk, new\_op\_pk, epoch, valid\_from, \sigma_{\text{root}}\}$ with strictly monotone epochs; (3) the cert rides in a block; on commit every node applies it and the owner swaps its live signer in lockstep — demonstrated in development with zero missed blocks; (4) the engine is the **single consensus signer** per node, its monotone *used* $\|$ *state* fsynced *before* any signature releases (reserve-then-sign). A real leaf-reuse bug found by treating the devnet as mainnet was fixed and disclosed — part of the verification culture (§12).

5.4 Signature bandwidth

Hash-based votes are kilobytes. At scale the answer is structural: **STARK-compressed quorum certificates** — a checkpoint proof that a supermajority signed — so light clients verify one recursive proof (§16, P4).

6 The transparent skeleton

6.1 Accounts

An account is an identifier, balances, a nonce, and a hash commitment to its current authentication key. Transactions open the commitment, sign payload plus next-key commitment, and advance the nonce — one-time-ness as a ledger property.

6.2 MandateTree v2

The network's signature primitive: a tree of budget envelopes enforced by validators at admission. Each mandate carries a holder, an asset, a drip rate (budget accrues continuously to `buffer_max` — allowances refill, not reset), a per-transaction cap, an expiry, and a parent. Semantics:

- **Attenuation at creation**: a child's per-transaction cap never exceeds its parent's.
- **Deliberate oversubscription, global enforcement**: a $45 root may promise children $65; every spend walks the *entire ancestor chain*, and the first level with insufficient buffer refuses. Delegation is a promise; the envelope is the law.
- **Org pays, agent authorizes**: funds move from the root's funding account; authority is the leaf holder's signature.
- **Two-phase atomicity**: authorization strictly precedes effects; refusal never half-mutates. The refusal is a typed receipt: `rejected: mandate: insufficient buffer at depth 1 from leaf (have 5000000, need 10000000)`.
- **Cascade revocation**: revoking a node severs every descendant instantly.

6.3 QHT channels

A channel opens under a mandate leaf (escrow = `unit_price`×`max_steps`, drawn through the ancestor walk once, at funding), anchored by a PayWord chain tip. Each micropayment reveals the next preimage — one 32-byte word, one hash to verify, no signatures in the hot loop. Settlement is proof-carrying (payee or watchtower submits the deepest word; one transaction pays *steps* × *unit_price*); expiry refunds the payer. Measured demonstration: **1,000 metered API calls settled in a single transaction**; a facilitator ran a real per-query retrieval service on this loop. Channels are the throughput multiplier of §13; `max_steps` is 32-bit, and hypertree channels are the designed long-lived upgrade.

7 The shielded pool

7.1 Notes, tree, anchors, nullifiers

A note is (v, tag, ρ, rcm) with on-chain form $cm = H_{\text{note}}(v \,\|\, tag \,\|\, \rho \,\|\, rcm)$. Commitments append to a depth-32 Merkle tree in frontier form ($O(32)$ hashes and storage per insert). Every block seals the root as an **anchor**; a 128-block window admits proofs against recent roots. Spending publishes the **nullifier**

$$nf \;=\; H_{\text{nf}}(nk \,\|\, \rho),$$

with nk the owner's *secret* nullifier key: unlinkable to its commitment, and — because nk never leaves the owner — **not even the sender can watch for the spend** (the sender knows ρ; without nk that knowledge is inert). The nullifier set is forever; conservation is public only in aggregate (`total_shielded`).

7.2 Transactions and the binding rule

MintToPool debits `value` transparently and appends cm; the mint proof attests cm opens to exactly `value` — the inflation guard. **ShieldedSpend** publishes $(anchor, nf, cm_1, cm_2, fee, credit?, mandate?)$ with value equation $v = v_1 + v_2 + fee$; $fee > 0$ with a credit account is the unshield channel, $fee = 0$ a fully shielded transfer. Spends are proof-carrying and relayable. Anti-malleability: the circuit commits to $tx\_binding = H_{\text{bind}}(credit \,\|\, fee)$ and the chain *derives the expected public statement from the transaction's own fields* before verifying — a relayer cannot redirect public effects without breaking the proof.

7.3 Addresses and spend trees

An address is (tag, kem_pk) with

$$tag \;=\; H_{\text{addr}}(spend\_root \,\|\, H_{\text{nk}}(nk)),$$

binding the root of a depth-10 Merkle tree over WOTS one-time public keys and the nullifier-key fingerprint. Wallets choose capacity $2^h \le 2^{10}$ with empty-ladder padding; addresses are cheap to rotate. In-circuit, the WOTS public key is *recovered from the signature* (chains completed; checksum blocks walk-forward forgery) — no key in the witness (∼85% witness reduction) — then folded up the spend tree to $spend\_root$ and checked against the note's owner tag. Addresses are public and reusable; each spend consumes a fresh one-time key *enforced by the statement itself*; payers cannot spend, track, or claw back. Lifetime addresses via an XMSS^{MT} hypertree are a measured-before-merged upgrade (D5).

7.4 The spend statement (circuit v3)

Public: $(anchor, nf, cm_1, cm_2, fee, tx\_binding)$. Witness: input note and Merkle path; WOTS signature and spend-tree path; nk; output notes. Enforced:

$$\begin{aligned}
cm_{\text{in}} &= H_{\text{note}}(v \| tag \| \rho \| rcm), & \text{Merkle}(cm_{\text{in}}, path) &= anchor,\\
wots_pk &= \text{Recover}(\sigma, d), & \text{Fold}(wots_pk, ots_path) &= spend_root,\\
tag &= H_{\text{addr}}(spend_root \| H_{\text{nk}}(nk)), & nf &= H_{\text{nf}}(nk \| \rho),\\
cm_1 &= H_{\text{note}}(out_1), \quad cm_2 = H_{\text{note}}(out_2), & v &= v_1 + v_2 + fee,\\
tx_binding &= H_{\text{bind}}(credit \| fee).
\end{aligned}$$

The same `no_std` circuit crate compiles into the prover guests *and* links into the chain's state machine as its hashing library — on-chain and in-circuit tree bytes cannot diverge (pinned by a keystone test).

8 The proving system

8.1 Stack, measured

The client prover is SP1 (RISC-V zkVM), selected by a bake-off of the *actual* statement across SP1, RISC Zero, and OpenVM. The journey: ∼26 minutes unoptimized → SHA-256 precompile (−60% cycles) → WOTS witness redesign (−56% more) → CUDA (611×) = **604,993 cycles, ∼1.24 s** (core, RTX 5090); production circuit v3 holds at 1.19–1.38 s. Agents prove their own payments; GPU-class hardware is a documented requirement.

8.2 Aggregation: one proof per block

An aggregator guest *verifies N compressed proofs inside the zkVM* and commits a digest binding each statement's kind, verifying key, and publics, plus order and count. The block carries the N transactions proof-less beside **one aggregate STARK**; validators verify it once and install per-block *coverage*: a proof-less transaction is valid iff the aggregate covered exactly the statement the chain derives from its fields. Coverage clears at block end; the per-proof path stays legal. Measured: **1,242 KB constant** regardless of N, ∼2.9 s at N=3 on one development GPU (∼25 KB per spend at N=50). Aggregation composes recursively (aggregates of aggregates), scaling near-linearly with proving hardware; in v1 the proposer aggregates, at scale a bonded, fee-earning, slashable **aggregator role** (D2).

8.3 Verifying-key pins and version pools

Genesis embeds SHAKE-256 pins of the spend, mint, and aggregator verifying keys; a node whose fetched keys mismatch **refuses to start**. Circuit evolution happens through **version pools**: a new statement or in-circuit hash arrives as a new pool version with fresh pins — an explicit, auditable upgrade event.

8.4 The in-circuit hash (D1)

Measurement chose SHA-256 in-circuit for pool v1: precompiled on the locked RISC-V prover at the numbers above, while arithmetization-friendly hashes pay an order-of-magnitude penalty there. *Hash-agility over hash-dogma*: version pools keep the choice rotatable if a STARK-native AIR tier changes the trade.

9 Confidentiality and discovery

Every output carries an advisory ciphertext: **ML-KEM-768** encapsulation to the recipient's per-epoch key (FIPS-203 deterministic), then the note sealed under **SHAKE-AEAD** (encrypt-then-MAC from domain-separated SHAKE-256; one-time keys; nonce bound to the commitment). Discovery is **trial decapsulation** with implicit rejection; on success the wallet *recomputes the commitment* against the on-chain leaf, so lying ciphertexts cannot plant phantom funds. The measured signature moment: a *fee* = 0 payment with zero transparent residue whose recipient reported `BOB DISCOVERED: $2 -- memo intact` while a third wallet found nothing. Consensus stores but never interprets ciphertexts (a bad blob hurts only its sender). KEM keys rotate per epoch while the address tag is epoch-stable — time-scoped visibility for free (§11); oblivious message retrieval is the researched successor at population scale.

10 Mandates × shielded

The defining composition: hierarchical caps, enforced by consensus, over balances nobody — including the enforcing validators — can see. A `ShieldedSpend` may reference a mandate leaf; the envelope sender must be the leaf's holder (the relayer *is* the authorizer); the mandate authorizes the spend's *public* component (the fee/unshield) through the full ancestor walk, authorization before effects, nullifier burned only on success. The reference sits outside the proof, so mandated spends aggregate like any other and the circuit is unchanged.

An organization shields its treasury once, deals allowances as stealth notes, and its envelope keeps the books over hidden state. In the live demonstration, agents a and b unshielded $20 each under a $45 root; agent c's $10 — allowed by its own $20 leaf — was refused by consensus at depth 1 with $5 remaining, its note untouched:

```
rejected: mandate: insufficient buffer at depth 1 from leaf
          (have 5000000, need 10000000)
```

Hidden-amounts mode (per-mandate committed accumulators, range-proved draw-downs, in-STARK) is specified and gated behind external circuit review (D3).

11 CVA: lawful disclosure without master keys

Three instruments, all holder-initiated, all cryptographically scoped:

- **One-time disclosure packages.** For exactly one payment: the note's one-time AEAD key, the ciphertext, the commitment's inclusion path, an anchor. Verification is a *pure offline function*: open the AEAD, recompute the commitment, fold the path, compare the anchor. Measured selectivity: the package key opened **0 of the 21 other ciphertexts** in the pool.
- **Epoch viewing keys.** One epoch's discovery+decryption: no spend authority, no nullifier key, no other epochs (measured: an epoch-1 payment stayed invisible to IVK(0)). Parent-mandate viewing keys extend the shape along the delegation tree.
- **CASP envelopes.** A registry on the transparent skeleton binds regulated providers to disclosure commitments; originator/beneficiary data travels as sealed, scoped attachments between the regulated parties (AMLR Art. 79 / IVMS-101), not into public state.

What does not exist: any master viewing key, any involuntary disclosure path, any governance mechanism able to create either (§14). The open problem — disclosure *completeness* — is handled by bonded attestations (stake slashed on contradiction) and is the network's flagship research program toward proof-carrying completeness. Regulatory posture: AMLR Article 79 applies 2027-07-01 and reaches "optional privacy features"; scoped disclosure is the *listing strategy*, and the network launched into that news cycle with the compliance architecture as the headline.

12 Node, wire, and engineering culture

12.1 The node

One Rust workspace: deterministic state machine (accounts, mandates, channels, pool) with typed receipts; the Malachite engine with the hash-based signing provider; in-node STARK verification against pinned keys; mempool and RPC; per-block receipt logs. The proof interface defaults to **RejectAll**: an unwired verifier refuses shielded traffic rather than trusting it. Seventy-six tests at the mainnet-candidate snapshot include keystone chain↔circuit consistency, two-node determinism replays, and codec roundtrips.

12.2 The wire

Consensus wire and WAL are binary (bincode); byte fields are format-aware (hex in human surfaces, raw on the wire), so a 2.7 MB proof costs ∼2.7 MB — a measured 4× JSON tax in development motivated the change. Two hashes are codec-independent by construction: transaction signing digests/txids (canonical JSON of fixed-field structs) and consensus sign-bytes (manual layouts). **No signature domain can depend on the wire format.**

12.3 Demo-gated engineering

Nothing counts as done until it runs live. Every increment shipped with a scripted demonstration including adversarial steps — double spends, forged proofs, lying ciphertexts, over-cap spends — refused with named receipts. Found bugs were fixed *and disclosed*. The honesty ledger is a maintained artifact.

13 Performance and capacity

13.1 Measured envelope

Metric	Value
Block cadence (4 validators, hash-signed votes)	∼1.4 s, round 0
Client spend proof (core, RTX 5090)	1.19–1.38 s
Mint proof	∼1.05–1.19 s
Compressed proof (recursion input)	1.8–2.2 s · 1,242 KB
Aggregate STARK (any N)	**1,242 KB constant · ∼2.9 s @ N=3**
In-node verify	∼87 ms/proof; ONE aggregate/block
Shielded tx, proof-less (with stealth cts)	∼2.7 KB
Proof on the binary wire	∼1× raw size

13.2 The capacity model, tiered

Tier 1 — configured chain ceiling: ∼183 TPS (256 tx/block ÷ 1.4 s; both deliberate constants, not physics). **Tier 2 — per-proof fallback: ∼8–18 TPS** (MB-scale proofs against wire and serial-verify budgets; exists as the legal fallback). **Tier 3 — aggregated: chain-side returns to the batch cap** (256×2.7 KB + 1.24 MB ≈ 1.9 MB/block; one verify; ∼64 hashes/tx of state work); the binding constraint moves to aggregator proving, which recursion trees scale with GPUs, while client proving is distributed by construction. **Tier 4 — configuration lifted: thousands of TPS, unproven** (gossip ≈27 MB/s at 10k shielded TPS; apply rate; WAN pacing; aggregator farm — exactly what pre-mainnet load benches measured). **Tier 5 — effective payments: ∼1,000× the settle rate** via channels — **∼183,000 payments/s** at the development configuration, with channel depth (32-bit steps) as the multiplier.

13.3 The throughput program ("the 183k plan")

Staged and demo-gated: **C1** measure (storm harness; the aggregation scaling curve N=10 . . . 256; apply-rate and WAN floors → the capacity sheet), **C2** lift configuration (batch 1024+, ∼1 s pacing, parallel fallback verify, admission pre-checks), **C3** aggregation at scale (tree-of-aggregates; multi-GPU; the bonded aggregator market), **C4** channel depth (10k-step ladders; watchtower defaults; hypertree channels), **C5** state-machine scaling only if C1 proves it binding (merkleized commitment; parallel disjoint-nullifier lanes — such spends commute). Milestones: M1 sustained 183 TPS; M2 full blocks under one root aggregate; M3 **≥183,000 effective payments/s demonstrated**; M4 1M+/s at lifted configuration. M1–M3 require no protocol changes.

14 Economics and governance

14.1 Money on the chain

Stablecoins are the product currency; $HKN is the security and work token: **staking** (validators bond; delegation extends; equivocation — including one-time-leaf reuse — is slashable evidence), **surety bonds** (organizations post slashable $HKN behind agent-fleet envelopes), **aggregator work** (the aggregation role is bonded and fee-earning), and **fees** (aggregate economics keep verification flat as volume grows). Revenue arrives in order: settlement fees and facilitator take, then bonds, staking economics, institutional envelope integrations. Illustrative economics are labeled as such; the network claims no revenue it has not settled.

14.2 Parameters vs constitution

Governable: fee schedules; staking/bond minimums; slashing ratios; aggregator-market parameters; envelope defaults (depths, windows, epochs); treasury and grants; circuit/pool upgrades via version pools with fresh pins. **Constitutional (not up for vote)**: the six doctrines of §2. No vote can introduce a pairing wrap, admit a lattice signature to a money path, create a master viewing key or involuntary disclosure, or flip the network transparent. Decentralization posture is stated plainly at every stage; the mainnet path ran founder devnet → public incentivized testnet → post-audit mainnet with rotation-native validator operations.

15 Security analysis

15.1 Threat model against a quantum adversary

Surface	Primitive	Quantum posture
Consensus votes	LMS/HSS over SHAKE-256	Hash-based; Grover answered by parameters
Validator identity	SLH-DSA-SHAKE-192s	Stateless hash-based; category 3
Account/spend authority	WOTS + spend trees, consensus-guarded	Hash-based; one-time enforced in-circuit
Channel payments	PayWord chains	Hash preimage
Spend validity	STARK (FRI/hashes; no pairings, F1)	Post-quantum sound
Note confidentiality	ML-KEM-768 + SHAKE-AEAD	Lattice KEM: break leaks metadata, never funds (D-2)
Transport identity	Ed25519 (libp2p only)	Classical; endpoint impersonation only

Harvest-now-decrypt-later touches only the confidentiality row; D-2 bounds its blast radius to privacy, and epoch keys bound any single compromise's window.

15.2 Protocol-level defenses

Double spends: the forever nullifier set. Forgery: raw-STARK soundness plus byte-exact, *chain-derived* public statements. Malleability: the binding rule. Inflation: mint proofs plus the public conservation ledger. Divergence: consensus-fatal value-id binding. Stateful-signature hazards: consensus rules plus reserve-then-sign. Lying ciphertexts: re-commitment at scan. Aggregate smuggling: digest-bound kind/key/order/count with chain-derived coverage. Domain confusion: codec-independent signing digests; every hash domain-tagged.

15.3 Assurance

Three audit tracks (consensus + state machine; STARK circuits; the stateful-key manager) gate mainnet-critical claims, with trust boundaries, a crypto inventory with KAT status, and eight consensus-critical invariants pre-mapped. Public residuals: GPU-class hardware assumed for agent proving; development numbers are single-machine WSL2; in-circuit WOTS awaits its full KAT campaign; disclosure completeness is open and stated as such.

16 The road here: phases and gates

P0 — the fundable demo (complete 2026-08-16). Sovereign 4-validator devnet; MandateTree and channels native; the $50 storyline as real transactions — including the canonical refused overspend — and 1,000 micropayments settled in one transaction; a working paid-search facilitator. *Gate G0 passed on the live demo.*

P1 — quantum-secure core (complete 2026-08-17). Consensus votes swapped to LM-S/HSS live; single signer + reserve-then-sign after a found-and-disclosed leaf-reuse bug; SLH-DSA roots and live mid-chain rotation. The bake-off produced F1, the locked SP1 stack, and the 1.24 s proof. *Gates G1, G2 passed on measurement.*

P2 — the private bank opens (complete 2026-08-17). The pool as consensus state; real STARKs verified in-node; stealth payments with scanning discovery; offline disclosure and epoch IVKs; the aggregation tier live; mandates composed with the pool; binary wire; vk-pinned genesis. Every increment demo-gated.

P3 — hardening to mainnet. Public incentivized testnet (external validators; 30-day incident-free soak; load benches producing the capacity sheet); three audits; CASP envelope registry and IVMS-101 binding; stablecoin issuance path; TGE; **mainnet launch, shielded by default, into the AMLR Article 79 window**.

P4 — scale and moats (the live roadmap). Oblivious message retrieval; STARK-compressed quorum certificates; hidden-structure MandateTrees and hidden-amounts mandates (D3) behind review; disclosure-completeness proofs v1; private KYA credentials; ISO 20022 pilots only with signed banking partners; an own-issuer stablecoin track.

The founding plan budgeted roughly eleven months for P0–P2; the demo-gated build delivered them in three days of founder+AI execution — documented, test-covered, and disclosed. The discipline that built the network is the discipline that runs it.

17 Related work

Zcash/Orchard pioneered shielded pools and viewing keys; its own process concedes non-post-quantum commitment bindingness, and its disclosure story stops short of offline-verifiable single-payment packages. Penumbra contributed the tiered-commitment-tree pattern this design's frontier tree simplifies from. Monero demonstrates the delisting cost of unscopeable privacy. Ethereum's Lean roadmap validates the hash+STARK endgame — years out; HashKinetics launched there. x402/AP2/ERC-8004 define agent payment, mandate, and identity envelopes at the application layer; HashKinetics is settlement beneath them and speaks their formats at its edges. Processor- and dashboard-level agent caps are the foil: HashKinetics puts caps in consensus. Private-compute networks offer privacy without post-quantum settlement. The intersection — PQ settlement × native shielded balances × consensus-enforced hierarchy — is this network's position.

18 Conclusion

HashKinetics exists because the agent economy needs what neither transparent chains nor classical privacy coins nor app-layer guardrails provide: an allowance, not a vault key — enforced by the ledger, invisible to competitors, disclosable to the law, and secure against the adversary that arrives with a quantum computer. The answer is conservative cryptography composed ambitiously: hashes for authority, a lattice for nothing but secrecy, STARKs for truth, recursion for scale, hierarchy for control, scoped disclosure for legitimacy. Every load-bearing claim traces to a measured demonstration or a named, gated plan.

The chain refused an overspend it could not see. Everything else follows from taking that sentence seriously.

References

[1] NIST FIPS 205, *Stateless Hash-Based Digital Signature Standard* (SLH-DSA), 2024.

[2] NIST FIPS 203, *Module-Lattice-Based Key-Encapsulation Mechanism Standard* (ML-KEM), 2024.

[3] RFC 8554, *Leighton–Micali Hash-Based Signatures* (LMS/HSS), 2019.

[4] NIST SP 800-208, *Recommendation for Stateful Hash-Based Signature Schemes*, 2020.

[5] Malachite BFT engine (Informal Systems), Apache-2.0; production validation incl. Circle's Arc (2026 reporting).

[6] SP1 zkVM (Succinct) v6.x; RISC Zero; OpenVM — bake-off subjects, `zkvm-bakeoff/RESULTS.md`.

[7] x402 Foundation launch and membership (2026-07); IETF draft-vauban-x402 (post-quantum receipts); Google AP2; ERC-8004.

[8] EU Anti-Money Laundering Regulation, Art. 79 (application 2027-07-01).

[9] Zcash ZIP 2005 (post-quantum considerations; commitment bindingness).

[10] Penumbra protocol documentation (tiered commitment tree).

[11] Rivest and Shamir, *PayWord and MicroMint*, 1996.

[12] Ethereum Foundation, Lean roadmap materials (hash-based signatures + STARK direction).

[13] HashKinetics repository: `HASHKINETICS-IMPLEMENTATION-PLAN.md`; `docs/MASTER-BUILD-PLAN.md`; `docs/SHIELDED-POOL-SPEC.md`; `docs/TECHNOTE-CONSENSUS-GUARDED-STATEFUL-HASH-SIGNATURES.md`; `docs/AUDIT-SCOPE.md`; `zkvm-bakeoff/RESULTS.md`; research reports 01–03 (2026-08-15, sourced and dated).