Version 0.2 — May 2026
Pyde Network · Apache-2.0

Pyde: A Post-Quantum, MEV-Resistant Layer 1 with DAG Consensus

A fairer, future-proof Layer 1, designed for the next decade of crypto.

Abstract

Pyde is a Layer 1 blockchain built greenfield to ship four properties as defaults from genesis. The four are technical commitments; each translates into concrete outcomes for the businesses building on Pyde, the developers writing for it, and the users transacting on it.

1. **Post-quantum cryptography by default.** FALCON-512 signatures, Kyber-768 threshold encryption, Poseidon2 + Blake3 hybrid hashing. No pre-quantum primitive on any consensus or account path. For businesses, long-tail agreements — insurance policies, multi-year escrows, intellectual-property registries, legal records — remain cryptographically valid into the quantum era without a coordinated migration to budget for. For users, funds remain secure as quantum computing matures.
2. **MEV resistance at the protocol layer, not via a trusted relayer.** Threshold-encrypted mempool + commit-before-reveal ordering + DAG consensus. Sandwich attacks, front-running, and proposer extraction are not policed or auctioned; they are structurally impossible. For users, this means trades execute at the price signed; for businesses, no invisible tax on customer transactions and no third-party relayer to opt into and trust.
3. **Sub-second finality.** Mysticeti-style DAG consensus, ~500 ms median commit finality, an 85-of-128 FALCON quorum certificate. For users, transactions confirm immediately rather than after a 12-second spinner; for businesses, settlement completes before checkout abandonment kicks in, and every confirmed transaction carries a portable cryptographic receipt that compliance teams verify offline.
4. **Commodity-hardware decentralization.** Full nodes and validators awaiting committee selection run on 8 cores / 16 GB RAM. Validators on the active committee at production throughput require a 500 Mbps – 1 Gbps NIC; every committee seat carries one vote regardless of stake. Enterprises that want to verify the chain independently can do so at hardware costs measured in thousands per year — not the $20K+/month that production-grade validators on the highest-throughput chains run.

The execution layer is **WebAssembly via wasmtime** (with Cranelift AOT) and a hybrid parallel scheduler that combines static access lists (Solana-style) with optimistic Block-STM speculation (Aptos-style). Developers author smart contracts in Rust, AssemblyScript, Go (TinyGo), or C/C++ — any wasm32-target language — with Pyde safety attributes (reentrancy off by default, checked arithmetic, typed storage, no `tx.origin`, compile-time access-list inference) preserved as language-native attributes and enforced at runtime. No proprietary VM or new language to learn; teams use the stack they already know. The `otigen` developer toolchain handles project scaffolding, build, state binding generation, and deployment. Cross-chain interactions are served by the **parachain framework** (v1) and post-mainnet bridge contracts gated by `HardFinalityCert` — a FALCON quorum certificate verifiable on any chain. The parachain framework lets developer teams launch their own execution environments (custom VMs, confidential-vote chains, gaming-specific subchains, oracle networks) without slot auctions or central gatekeeping; Pyde validators stake to run third-party parachains and earn the parachain's fees.

This document presents the current design following a **May 2026 architectural pivot** from an in-house HotStuff variant (whose persistent wedges and stalls at 400 ms slot timing motivated a clean rebuild) to a DAG-based consensus inspired by Narwhal, Bullshark, and Mysticeti. The pivot scoped the chain to its execution and cryptography layers first; the consensus layer is being rebuilt design-first against the new foundation.

The v1 mainnet throughput target — for both the plaintext and encrypted regimes, on commodity validator hardware — is established by a multi-region performance harness before any number is published. Long-term aspirational headroom (with GPU acceleration, batch threshold decryption, and protocol upgrades) is real but carries no concrete number and is not a v1 commitment. The chain commits to publishing only what the harness measures under sustained, production-realistic conditions — never lab extrapolations or microbenchmark peaks — so any number it eventually publishes is one application teams and businesses can plan against rather than aspire to.

---

1. The Problem

Four open architectural debts run across the production L1 set today, each of them protocol-level rather than application-level, and each easier to ship at genesis than to migrate into a chain that has been running without it. The debts are paid by different parties — users lose to MEV; businesses absorb the cost of unpredictable fees and delayed settlement; developers carry the integration burden of bridge multisigs and proprietary languages — but they all trace to the same handful of design choices.

Quantum vulnerability. Every major L1 in production — Bitcoin, Ethereum, Solana, Cardano, Polkadot, Aptos, Sui — secures its consensus and account paths with classical cryptography (secp256k1, Ed25519, BLS12-381) that falls to Shor's algorithm on a cryptographically-relevant quantum computer. NIST's 2024 standardization of FALCON, ML-DSA, and ML-KEM unblocked the post-quantum primitives, but retrofitting them into a live chain with trillions of dollars at risk and deployed contracts hard-coded against pre-quantum key formats is a multi-year coordinated migration. The chains have not been blind to the problem; the constraint is the shape of the migration, not the seriousness of the response. Long-tail contracts written today on those chains carry an unbudgeted migration risk: insurance policies, multi-year escrows, intellectual-property registries, and other legal records that assume cryptographic validity over decades inherit whatever migration plan the host chain eventually executes.

MEV extraction. Maximum Extractable Value has hardened into a multi-billion-dollar tax paid by retail users to validator-builder coalitions. Sandwich attacks, front-running, and proposer extraction are not bugs to be patched — they are structural consequences of public mempools combined with single-proposer block production. The incumbent response has been to make the MEV market more efficient (proposer-builder separation on Ethereum, Jito on Solana). The alternative — removing the information asymmetry at the protocol level — is harder to retrofit because builder economics are now baked into the validator revenue model. Businesses building exchange, swap, or trading infrastructure on those chains either accept the tax on their customers or take on an opt-in dependency on a third-party relayer they must trust to behave.

Throughput at finality. Chains optimizing hardest for throughput have made validation a premium-hosting business. A Solana validator at production performance requires 12+ cores and 256+ GB RAM, costing $20K+/month. Chains optimizing for decentralization have ended up with throughput unusable for serious applications. The combination — sub-second hard finality at retail-scale throughput on commodity hardware — is the category no production chain occupies cleanly today. The trade-off lands on users (twelve-second confirmations, abandoned checkouts), on businesses (settlement delays, cash-flow friction, support tickets about transactions in flight), and on enterprises that wanted to verify the chain independently but were priced out of running their own validator.

Centralization at scale. Validation, smart-contract deployment, and cross-chain interaction have all converged toward gated infrastructure: data-center validators, custodial bridge multisigs ($3B+ lost in bridge hacks since 2021), oracle networks run by small operator coalitions, app-chain slots auctioned to well-capitalized parachain teams. Each is a coherent local response to the constraint set the chain in question faced; the cumulative cost lands on users (forced trust in operators they did not choose) and on developer teams (wanting to launch their own execution environment but unable to afford the slot auction or denied a place on the team's shortlist).

These four problems are not independent items. They converge in time. NIST's 2024 standardization matured the cryptographic primitives at the same moment that MEV literature converted into quantified user-cost numbers, at the same moment that Solana's hardware creep made the decentralization cost visible, at the same moment that L2 sequencer trust assumptions started attracting public scrutiny. The architecture that wins the next decade does not have to be the one that won the last one. **No chain in production today provides all four properties as defaults.** Pyde is the chain built to occupy that position.

---

2. Four Axioms

Every design choice in this document follows from four axioms.

Axiom 1 — Post-quantum cryptography is the default. No application-layer signature, encryption, or hash in Pyde uses pre-quantum primitives. FALCON-512 signs every consensus vote, every transaction, every validator key

registration. Kyber-768 / ML-KEM encrypts every encrypted-mempool transaction. Poseidon2 (Goldilocks field) hashes ZK-bearing commitments; Blake3 hashes the high-volume native paths where ZK-friendliness is not in scope. Ed25519 appears only in libp2p's noise transport for peer routing — a quantum attacker who breaks Ed25519 learns the network topology but cannot forge a vertex, decrypt a transaction, or compromise an account.

The trade-off is signature size: 666 bytes for FALCON-512 versus 64 bytes for Ed25519, 1,088 bytes per Kyber-768 ciphertext versus negligible plaintext overhead. Pyde absorbs that cost in the layers that matter and avoids it everywhere it does not (e.g., gossip-level message authentication uses Blake3 + libp2p noise). For users, this means funds and identities remain secure as quantum hardware matures; for the ecosystem, long-tail agreements signed on Pyde — insurance policies, multi-year escrows, intellectual-property registries, legal records — don't carry an unbudgeted future-migration cost.

Axiom 2 — MEV is a protocol bug. No committee validator must be able to read, reorder, or selectively include unconfirmed transactions. This is a security property, not a market-design problem. Pyde achieves it with three interlocking mechanisms (Section 8 has the details):

1. Transactions can be encrypted under a Kyber-768 threshold public key held jointly by the 128-validator committee — no fewer than 85 of 128 shares can decrypt.
2. The committee commits to a canonical order at the DAG anchor before any decryption share is released. The order is fixed by the time content is visible.
3. There is no single proposer. Order emerges deterministically from the DAG by every honest validator independently; no committee member can reorder, exclude, or front-run.

The combination removes the surface MEV extraction needs to exist on. Encryption is opt-in per transaction; simple transfers go plaintext for lower fees, MEV-sensitive operations (DEX swaps, NFT mints, liquidations) opt into encryption. For applications building exchange, swap, or trading infrastructure, this removes the choice between accepting a hidden tax on customer trades and opting into a third-party relayer they must trust to behave.

Axiom 3 — Throughput requires parallel execution in a single binary. Consensus and execution share a single process. The execution layer is a WebAssembly execution (wasmtime + Cranelift AOT) with a hybrid parallel scheduler: static access lists for functions with compile-time-known accesses, Block-STM speculation for dynamic accesses. The choice is monolithic over modular: every cross-layer boundary is a trust boundary and a latency cost; for an L1 whose target is high-throughput low-latency MEV-resistant execution, coherence is worth more than heterogeneity. Cross-chain interoperability is added back as a separate permissionless parachain layer above the coherent base, not as a structural premise that fragments the chain at genesis. For investors evaluating execution-layer maturity, the monolithic-binary choice means one operational surface — one team's runbook, one set of audits, one performance harness — rather than the coordination cost of a microservices-style L1.

Axiom 4 — Decentralization is the protocol's burden, not the user's. Validators run on commodity hardware. Every committee member has exactly one vote regardless of stake — the validator bond is anti-Sybil cost, not a power multiplier. Cross-chain infrastructure is permissionless: any operator who stakes PYDE and runs a Pyde-published spec joins the parachain operator set, no auctioned slots, no gatekeeping team. The cost of participating in Pyde — running a node, validating, building a parachain — is a function of will and a small fixed bond, not access to data-center capital or auction proceeds. For developer teams wanting to launch their own execution environment, that means no slot auction to win and no foundation shortlist to make; for enterprises wanting to verify Pyde independently, the validator hardware sits comfortably inside the IT budget.

---

3. The May 2026 Pivot

Pyde's earlier architecture used an in-house pipelined HotStuff variant with VRF proposer selection at 400 ms slot timing. Repeated wedges — head-divergence deadlocks, view-change cascades, quorum starvation under network jitter — were being addressed by accumulating patches rather than fundamental changes. The team made a clean break: remove the entire consensus, mempool, and networking layers from the active workspace and rebuild against a foundation with a smaller protocol surface and simpler safety arguments.

Post-pivot:

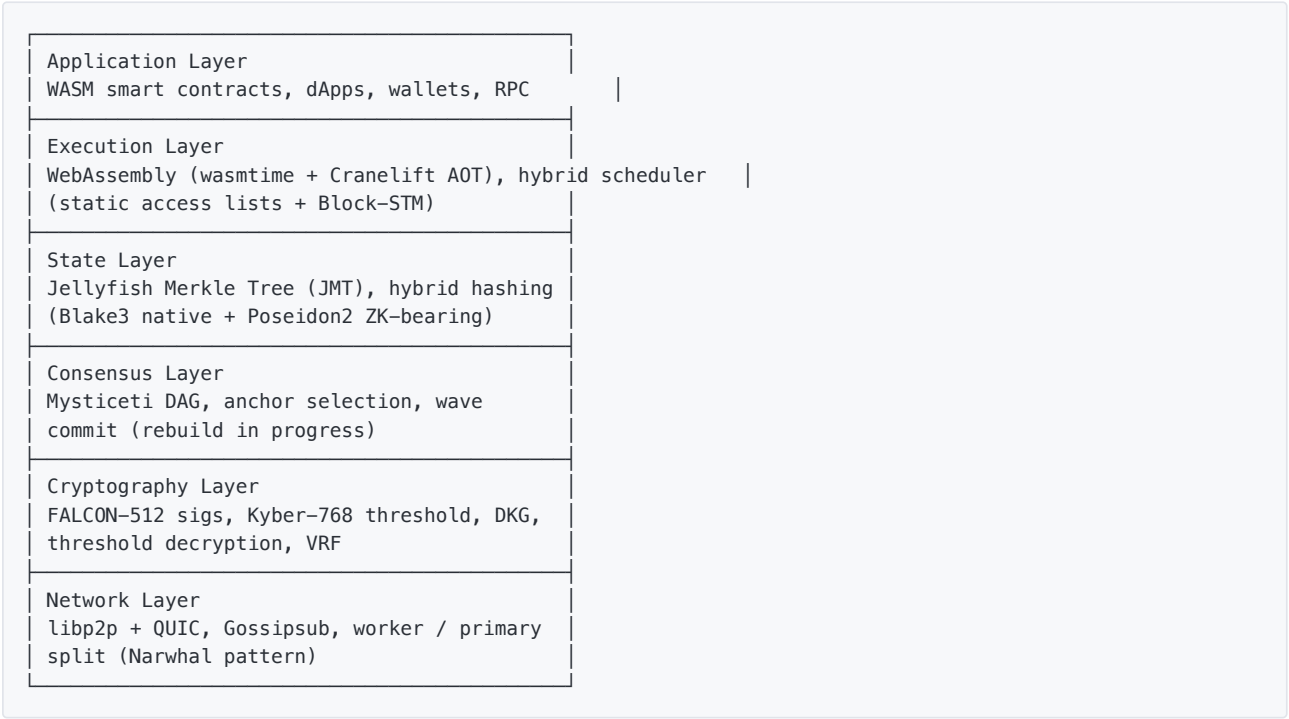
- The active engine workspace contains five execution-layer crates: `crypto`, `execution` (the wasmtime-based WASM executor), `state`, `account`, `tx`. Nothing else.
- Consensus, mempool, networking, slashing, and the node binary have been moved to a `archive/` archive for reference.
- The next consensus layer is being designed against the lessons of HotStuff failure: no view changes, no single-proposer bottleneck, data-driven round advancement, structural censorship resistance.

The decision: **Mysticeti-style DAG consensus**, with FALCON-bound vertex production and threshold-decryption ceremonies pipelined into the commit boundary. The remainder of this document describes the post-pivot design.

---

4. Architecture

Pyde is a monolithic Layer 1 chain — consensus, execution, and state in a single binary — with a layered protocol structure.



Three operational tiers run the same binary; role differentiation is configuration:

Tier	Stake	Committee role	Earns
Validator	10K PYDE min (single tier)	Eligible for uniform-random committee selection each epoch	Reward-pool share (stake × uptime) + inflation share + activity-weighted bonus while on active committee
RPC node / full node	None	None	Off-chain RPC fees (market-set)

---

5. Cryptography

5.1 FALCON-512 Signatures

Every transaction, vertex, and state-root attestation is signed with FALCON-512 (NIST FIPS 206). Properties:

- Signature size: ~666 bytes (variable, hard cap 1,280 bytes). Public key: 897 bytes.
- Verification: ~80 µs on commodity x86_64 / ARM64.
- No post-quantum BLS analog has matured, so consensus quorum certificates are the union of N FALCON signatures over a `voter_bitmap` rather than a single aggregated signature. The mainnet bandwidth budget (500 Mbps – 1 Gbps NIC at the relevant TPS tier) is sized to absorb the QC size.

5.2 Kyber-768 Threshold Encryption

Pyde's encrypted mempool uses Kyber-768 (NIST FIPS 203) with a threshold variant. At each epoch the 128 committee members run a Distributed Key Generation ceremony producing one public key `PK` and 128 shares `s_i`. The threshold is `2f + 1 = 85` of 128 — the same quorum that gates commit and finality.

Transactions can optionally be encrypted under `PK` before submission. Decryption requires 85 committee members to compute partial decryptions and combine them by Lagrange interpolation. **No coalition of fewer than 85 can decrypt anything** — the unique secret only exists in distributed form.

Critical invariant — commit-before-reveal. Consensus commits to an order at the DAG anchor before any decryption share is released. By the time content is revealed, the order is fixed and irreversible. This is what eliminates MEV at the protocol layer.

5.3 Hybrid Hashing: Blake3 + Poseidon2

Use	Hash	Reason
JMT internal nodes (high volume)	**Blake3**	~30× faster than Poseidon2 on CPU; not in ZK circuits
Published state root (per commit)	**Both** (Blake3 native + Poseidon2 ZK)	Native verification fast; ZK validity proofs future-compatible
Transaction hashes — ciphertext	Blake3	Gossip / dedup, not in ZK
Transaction hashes — plaintext canonical	Poseidon2	Inside sig-verify ZK circuits
Address derivation	Poseidon2	`Poseidon2(falcon_pk)` exposed to sig-verify circuits
FALCON sig payload hashing	Poseidon2	Inside ZK aggregation

Poseidon2 over the Goldilocks field is the algebraic hash everywhere a future ZK proof would need to re-derive the value in-circuit; Blake3 is the high-throughput native primitive where ZK exposure is not in scope.

5.4 Randomness Beacon

Each epoch's beacon is produced by the previous epoch's committee via a threshold-signature ceremony on a known message. ≥ 85 shares combine into a deterministic aggregated signature; the hash of the signature is the beacon, 32 bytes. The beacon seeds:

- Per-round anchor selection: `anchor_member_id = Hash(beacon, round, prev_state_root) mod 128`
- Next epoch's committee VRF picks
- Other protocol randomness

The `prev_state_root` term reduces anchor predictability from a full epoch (~ 3 hours) to a few rounds (~ 450 ms).

---

6. Consensus: Mysticeti-Style DAG

6.1 Why DAG (Why Not HotStuff)

The pre-pivot HotStuff variant exhibited persistent wedges and view-change cascades under realistic network conditions. The DAG approach removes the fragile parts:

Problem in HotStuff	DAG resolution
Single proposer bottleneck	No proposer — every member contributes vertices each round
View-change protocol complexity	No view changes — eliminated an entire failure class
Timing-driven slot pipeline	Data-driven rounds advance with quorum, not clock
Proposer can selectively censor	127 honest members can include any tx; censorship requires near-unanimous collusion
Throughput limited by leader bandwidth	Throughput scales with committee size

The DAG also integrates cleanly with threshold decryption: the commit boundary is the natural place to run the decryption ceremony, with partial shares piggybacked on vertices in the rounds leading up to it.

6.2 Worker / Primary Split (Narwhal Pattern)

Each validator runs two roles:

- **Workers** (N processes): handle transaction ingress, build batches, gossip batches to peer workers.
- **Primary** (one process): handles consensus — produces vertices each round, gathers parent references, signs state roots, runs the DKG.

Transactions traverse the network once via worker gossip; consensus vertices stay tiny because they carry only batch hashes by reference (Section 6.3).

6.3 The Vertex

Each round, every committee member's primary produces exactly one vertex:

```
struct Vertex {
    round: u64,
    member_id: u32,
    batch_refs: Vec<BatchHash>,                 // hashes of batches I have
    parent_vertex_refs: Vec<VertexHash>,        // ≥ 85 round-(N-1) hashes
    state_root_sigs: Vec<StateRootSig>,         // attestations on recent commits
    prev_anchor_attestation: VertexHash,        // attestation of prior anchor
    decryption_shares: Vec<DecryptionShare>,    // piggybacked partials
    falcon_sig: FalconSig,                      // sig over the vertex
}
```

Parents must come strictly from the prior round (no skip edges in v1). The DAG is a consensus structure; transaction data lives in batches stored at the worker layer, referenced by hash.

Vertex size: typically ~ 830 bytes minimal, ~ 25 KB heavy (50 batches + 5 state-root sigs + 85 decryption-share partials); hard cap 64 KB.

6.4 Rounds, Anchor, and Commit

Rounds are data-driven: a member ticks from round N to N + 1 once it collects ≥ 85 valid round-N parents (the slowest 43 can lag without blocking anyone). Round rate: ~ 5–10 rounds / sec depending on network conditions.

Each round has a deterministically-selected anchor:

```
anchor_member_id = Hash(beacon, round, prev_state_root) mod 128
```

When the anchor vertex collects sufficient Mysticeti 3-stage support from later rounds, a commit fires:

1. Anchor's subdag is collected by walking `parent_vertex_refs` transitively.
2. The subdag is sorted deterministically: `(round, member_id, list_order)`.
3. Batches referenced by each vertex are dereferenced.
4. For each encrypted transaction within those batches (encryption is per-tx, not per-batch), the threshold decryption ceremony runs (pipelined — partial shares are already in flight by commit time via the `decryption_shares` field of vertices observed during the prior rounds).
5. wasmtime executes decrypted transactions in canonical order.
6. State root is computed (Blake3 + Poseidon2 dual), FALCON-signed by ≥ 85 committee members.
7. Finality is declared once ≥ 85 state-root signatures converge.

Median end-to-end finality target: ~ 500 ms. Validated by performance harness pre-publication.

6.5 Committee

128 validators per epoch, drawn from the global validator pool:

- **Selection:** uniform random from all validators with stake ≥ `MIN_VALIDATOR_STAKE` (10,000 PYDE). Single tier — no separate committee/non-committee stake floors.
- **Anti-Sybil:** operator identity binding, max 3 validators per operator.
- **Equal power:** every committee member has equal voting weight, equal vertex production rate, equal anchor probability. Stake influences only (a) eligibility and (b) the proportion of the flat 30 % stake-pool yield share. Activity rewards are contribution-weighted, not stake-weighted.
- **Epoch length:** ~ 3 hours wall-clock (round count varies with network conditions).
- **DKG:** runs in the background during the prior epoch's last minutes; the new committee has the threshold key ready by epoch start.

6.6 BFT Properties

For `n = 128`: `f = ⌊(n − 1) / 3⌋ = 42` is the maximum tolerable Byzantine count; the quorum threshold is `2f + 1 = 85`. This single number appears throughout the protocol (vertex certification, commit support, threshold decryption, state-root sigs, DKG output) — consistency across uses avoids attack edges from boundary mismatches.

Safety holds under any network conditions assuming at most `f = 42` Byzantine members. Liveness holds under partial synchrony.

6.7 Halts and Recovery

When safety appears at risk (e.g., contradictory state-root sigs), the protocol auto-halts. Three halt classes:

Class	Trigger	Authority
Soft stall	Network / quorum slack	Emergent (auto-recovers)
Hard halt	Contradictory state roots, equivocation cluster, DAG fork	Protocol-detected automatic
Emergency halt	Off-chain bug report, active exploit, hard-fork prep	Governance multisig (7-of-12)

Rollback is bounded to one epoch (~ 3 hours); within that window governance can authorize rollback to a prior consistent state. Beyond an epoch, only a coordinated hard fork is possible. This is the "weak finality with sunset" pattern — operational flexibility for early detection without arbitrary commit reversibility.

---

7. Execution: WebAssembly, Hybrid Scheduling

7.1 The WASM Execution Environment

Pyde executes smart contracts under **wasmtime**, the Bytecode Alliance's production WebAssembly runtime (in use at Microsoft, Fastly, Shopify):

- WebAssembly Core Specification: linear memory, structured control flow, validated bytecode, no syscalls
- **Cranelift AOT** compilation inside wasmtime — every module is compiled to native machine code at deploy time and cached; subsequent invocations re-use the compiled artifact, no JIT, no runtime recompile
- **Fuel-based gas metering** — every WASM instruction decrements a fuel counter at the basic-block level; when fuel hits zero, wasmtime traps and the transaction reverts
- **Per-instance sandbox** — each transaction runs in its own wasmtime instance with bounded linear memory (default 64 MB cap); the host (validator) decides which host functions are importable
- **Host Function ABI** for all chain interaction — storage (`sload` / `sstore` / `sdelete`), balance and transfers, crypto (Blake3, Poseidon2, Keccak256, FALCON verify, threshold encrypt/decrypt), events, cross-contract and cross-chain calls
- The retired Otigen language and custom `pyde-vm` register-based VM are preserved in the historical pivot record only; the active execution layer is wasmtime end-to-end

Determinism is load-bearing: the same WASM module with the same inputs and host-fn responses must produce byte-identical state transitions across all 128 committee members (consensus state-root agreement) and inside future ZK validity proofs over execution. Non-deterministic WASM features (threads, SIMD timing, floating-point environment) are disabled at module-validation time.

7.2 Smart Contract Authoring

Smart contracts are authored in **any wasm32-target language** (Rust, AssemblyScript, Go via TinyGo, C/C++). The `otigen` developer toolchain reads a `otigen.toml` , generates state bindings with pre-computed slot constants, invokes the correct language compiler, and produces a `.wasm` artifact plus JSON ABI:

- 30 keywords; storage maps, structs, enums, variable-length `Vec` , `String`
- 4-byte function selectors (EVM-compatible dispatch)
- Reentrancy guards (`#[reentrant]`), checked arithmetic by default, custom errors and events
- `#[view]` / `#[payable]` / `#[reentrant]` function attributes
- Compile-time static access-list inference: for each function, the compiler emits the set of storage slots it provably touches, plus regions where access depends on runtime values
- Block context is `block.anchor` (the wave's canonical anchor vertex), not `block.proposer` — Pyde's DAG has no single proposer, so contracts that depended on `block.proposer` on other chains do not have an analog here

7.3 Hybrid Parallel Scheduler

Two parallel-execution philosophies, used together:

- **Static access lists** (Solana-style): for functions where access is inferable at compile time, the scheduler partitions transactions into parallel groups by their declared access sets. Deterministic, no speculation overhead.
- **Block-STM speculation** (Aptos-style): for functions with dynamic access patterns, transactions execute optimistically. Read / write sets are tracked at runtime; conflicts trigger re-execution in canonical order.

The Build-time state binding generator emits both `declared_access_set` (static) and `dynamic_access_regions` (runtime). The runtime scheduler uses the static information for partition planning and falls back to Block-STM for dynamic regions. Pyde controls compiler, runtime, and language — making this hybrid feasible where most chains commit to one approach.

Preflight at user submission time (via `pyde_estimateAccess` RPC) returns a runtime-observed access list, which the wallet attaches to the transaction. The scheduler treats access lists as hints, verifying at runtime and falling back to

speculation on mismatch — safe by construction.

7.4 Transaction Lifecycle

```
Wallet:
  1. Construct tx (sender, recipient, amount, payload, ...)
  2. RPC pyde_estimateAccess → returns gas_estimate + access_list
  3. Attach access_list to tx
  4. FALCON-sign tx hash
  5. (Optional) Encrypt signed tx + access_list with epoch PK
  6. Submit to RPC

Worker:
  7. Validate wire format
  8. (Plaintext) verify FALCON sig at ingress
  9. Batch with other txs
  10. Gossip batch to peer workers

Primary:
  11. Produce vertex referencing available batches
  12. Gossip vertex; peers attest as parents in next round

Commit (~500 ms median):
  13. Anchor selected; subdag walked; canonical order emitted
  14. (Encrypted) threshold-decrypt batches
  15. wasmtime executes in canonical order
  16. State root computed, signed by ≥ 85 committee members
  17. Finality declared on 85 state-root sigs
```

End-to-end latency: ~ 500 ms median for plaintext, ~ 700 ms for encrypted (adds the decryption ceremony to the commit budget).

---

8. State: Jellyfish Merkle Tree

State is stored in a Jellyfish Merkle Tree (radix-16, path-compressed), persisted in RocksDB. Compared to a fixed-depth-256 Sparse Merkle Tree:

- ~ 5–10 nodes touched per state operation (vs ~ 256)
- Substantial I/O savings at high TPS
- Same authentication properties (Merkle commitment, inclusion / exclusion proofs)
- Production-proven (Diem, Aptos)

State commitment is dual-rooted at every commit: Blake3 for fast native verification by committee and validators, Poseidon2 for future ZK light clients and validity proofs. Both roots are signed by ≥ 85 committee members.

The block witness — every state slot touched by a wave plus a single batched JMT proof against the pre-state root — has a hard 1 MB cap, rejected at verification before any proof work runs.

---

9. MEV Resistance

Three structural defenses, layered:

Layer 1 — Threshold encryption. Users can encrypt transactions before submission. The encrypted blob is opaque even to committee members. Mempool sees only encrypted bytes; attackers cannot observe content to position around.

Layer 2 — Commit-before-reveal. Consensus orders encrypted transactions at the DAG anchor before any decryption share is released. By the time content is revealed, the order is fixed and irreversible.

Layer 3 — No proposer. Pyde's DAG consensus has no single party empowered to choose which transactions enter a commit or in what order. The canonical order emerges deterministically from the DAG; no committee member can selectively reorder, exclude, or front-run.

The combination eliminates the structural conditions for sandwich attacks, front-running, and proposer extraction. MEV is not policed or auctioned — it is structurally impossible at the protocol layer.

Encryption is opt-in. Simple transfers go plaintext for lower gas; MEV-sensitive operations opt in via `pyde_sendRawEncryptedTransaction`. Encryption adds ~ 200 ms to end-to-end latency (the threshold decryption ceremony).

---

10. Network Protocol

- **Transport:** QUIC over UDP, with TCP fallback. No head-of-line blocking, built-in TLS 1.3.
- **P2P library:** libp2p (Rust). Audited, used by Ethereum / Filecoin / Polkadot.
- **Node identity:** Ed25519 keypair (separate from validator FALCON key, rotatable).
- **Peer discovery:** layered (hardcoded seeds → DNS → on-chain validator registry → PEX → persistent cache). No DHT.
- **Gossip:** Gossipsub with per-topic meshes (`vertices` , `batches` , `decryption_shares` , `state_root_sigs` , `mempool` , `state_sync`). Message size limits per type, enforced at parse time.
- **DoS defense:** four layers — connection (IP / ASN caps), message (rate limits per type), peer scoring (misbehavior accumulates, decays with good behavior), application (gas tank prepayment for encrypted submission).
- **Committee defense:** sentry-node pattern (Cosmos-style) to insulate committee primaries from direct internet exposure.

Committee NIC requirement at v1's honest throughput target (to be established by the multi-region performance harness) is **≥500 Mbps**. Higher-throughput regimes (1 Gbps, 10 Gbps) appear in §12.1 below labeled as Stretch / Aspirational, not v1.

---

11. Cross-Chain: The Parachain Layer (Post-Mainnet)

Cross-chain interactions in Pyde — calling functions on other chains, querying oracles, requesting off-chain compute, indexing on-chain data — happen through a **parachain layer of permissionless decentralized infrastructure providers**. A parachain is *not* a sovereign app-chain. It is an open-source implementation of a Pyde-published specification, run by operators who stake PYDE, follow protocol-defined rules, and earn gas fees from contracts that call them.

11.1 The `cross_call!` Macro

```
cross_call!(
    target_chain = "ethereum",
    contract = "0x...",
    function = "balanceOf",
    args = [...],
    callback = "handle_balance_response",
);
```

The macro is asynchronous. The originating transaction marks the call pending and emits an event; the actual cross-chain or oracle work happens off-chain at the parachain operator set; the result arrives in a separate callback transaction.

11.2 HardFinalityCert

A FALCON quorum certificate over `(wave_id, blake3_state_root, poseidon2_state_root)`, signed by ≥ 85 of the active committee. Verification on any counterparty chain: 85 FALCON-512 verifies (~ 85 ms) plus a Merkle path — feasible on any chain with a reasonable VM. The cert's stability across the chain's lifetime is what makes parachains feasible without further protocol changes after mainnet.

11.3 Architecture vs Implementation

The protocol-level surface (the `cross_call!` macro, `HardFinalityCert`, unified gas model) is settled at genesis. The actual parachain layer — specification, reference implementations, operator economics, bridges to Ethereum / Cosmos / Solana — ships post-mainnet. The mainnet `cross_call!` initially returns a runtime "not yet supported"; contracts written today work without rewriting when parachains activate.

11.4 Why the Parachain Framework Is the Most Consequential Adoption Surface

The parachain framework is the chain's most consequential decision for ecosystem growth: third-party developer teams launch their own execution environments — custom VMs, confidential-vote chains, gaming-specific subchains, oracle networks, privacy-focused application chains — without an auction or foundation shortlist, while inheriting Pyde's security, sub-second finality, and `HardFinalityCert`-based composability for free. Each parachain bootstraps its own developer community around its specific innovation; Pyde validators stake to run third-party parachains and earn the parachain's fees. The model converts what Polkadot priced through slot auctions into a permissionless capability — and gives Pyde a structural ecosystem-growth path absent from monolithic single-execution chains. For investors, this is the line where Pyde's adoption story stops being one team's roadmap and starts being a many-team capability surface.

---

12. Performance

12.1 Honest Targets

The v1 mainnet throughput target is validated by a multi-region production-realistic harness before any number is published. Pyde publishes **no forward throughput figure**; latency targets and the hardware envelope, by contrast, are concrete:

Mode	v1	Stretch	Aspirational
Plaintext throughput (sustained, commodity)	awaiting harness	awaiting harness	awaiting harness
Encrypted throughput (sustained, commodity)	awaiting harness	awaiting harness	awaiting harness (GPU)
Median commit finality	~ 500 ms	~ 400 ms	~ 300 ms
Committee NIC	500 Mbps	1 Gbps	10 Gbps

The published throughput figure comes only from actual harness output, under the discipline of publishing only what the harness measures under sustained, production-realistic conditions — never lab extrapolations or microbenchmark peaks.

12.2 Hardware Tiers

Role	Hardware
Light client	Mobile / browser
Full node / RPC	8c / 16 GB / 500 GB NVMe / 100 Mbps
Non-committee validator	8c / 16 GB / 500 GB / 100 – 250 Mbps
Committee validator (v1 baseline)	8 – 16c / 32 GB / 1 TB SSD / 500 Mbps
Committee validator (Stretch, post-mainnet)	16c / 32 GB / 2 TB SSD / 1 Gbps
Committee validator (Aspirational, GPU-class)	32c / 64 GB / 4 TB SSD / 10 Gbps

The commodity-hardware promise applies layered: full nodes and validators awaiting committee selection stay on a developer workstation at every throughput level. The first committee row is the v1 hardware Pyde is sized against; the higher rows are post-mainnet scaling targets, not v1 commitments.

12.3 Methodology

Pyde's pre-pivot in-house HotStuff implementation measured ~ 4 K TPS in practice — well below the original 12,500 TPS design target it claimed. The lesson: **lab benchmarks ≠ production**. Pyde's performance discipline going forward:

- **Multi-region testing mandatory.** Localhost devnet numbers do not count.
- **Production-realistic workload mix.** Not synthetic transfer-only; realistic ratios of transfers / AMM swaps / NFT mints / contract calls.
- **Continuous soak.** 4-hour minimum for any TPS claim that ships externally.
- **Measured-only rule.** External claims publish only what the harness measures under sustained, production-realistic conditions — never lab extrapolations or microbenchmark peaks.
- **Public dashboard.** Rolling 30-day metrics, visible.

No TPS claim is published externally without harness evidence. This is non-negotiable, and it is the most important lesson absorbed from the pre-pivot reset.

12.4 What the Numbers Enable

The v1 throughput target at ~500 ms median finality on commodity hardware is sized to run a serious DEX, a settlement system, a high-frequency NFT marketplace, a payments rail, or a real-time gaming backend — without queueing or fee spikes during peak load. Application teams designing for Pyde plan against the harness-validated number, not an aspirational one; the chain's discipline is to publish what it can deliver and over-deliver as the harness validates higher tiers. For businesses sizing Pyde for production load, the contract is honest: the v1 figure is whatever the production-realistic harness has shown, not a marketing extrapolation.

---

13. Economics

13.1 Token

- **Total genesis supply:** 1,000,000,000 PYDE
- **Decimals:** 9 (1 PYDE = 10^9 quanta)
- **Inflation schedule:** 5 % year 1, decreasing to 3 % / 2 % / 1 %, fixed at 1 % thereafter

13.2 Validator Bonds

Tier	Minimum stake	Role
Validator	10,000 PYDE (single tier)	Eligible for uniform-random committee selection; 128 of the pool serve each epoch

Anti-Sybil cap: max 3 validators per operator (identity-bound). Bonding: 1 epoch before active. Unbonding: 30 days (must exceed the 21-day safety-evidence freshness window). Slashing applies during both bonded and unbonding states — preventing attack-then-exit.

13.3 Fee Model

EIP-1559 base fee with elastic 4 × blocks; **no priority tips**. Priority would re-introduce the information asymmetry the encrypted mempool eliminates, so it is structurally excluded rather than zeroed by policy. Every transaction pays exactly `gas_used × base_fee` — wallets quote a single number, not a range.

Each transaction's base fee splits deterministically:

- **70 % burned** (deflationary pressure)
- **10 % to treasury** (multisig-controlled, PIP-reviewed)
- **20 % to the reward pool**, distributed at epoch end:
- 70 % of the pool, activity-weighted across the active committee (vertices certified, batches included, decryption shares submitted, anchor selections)
- 30 % of the pool, flat across all staked validators (active-committee + awaiting-selection), distributed by stake × uptime

13.4 Indicative APY

Per-token yield is uniform across all validators (single tier; rewards distribute by stake × uptime). The activity-weighted committee bonus is layered on top during the ~3-hr epoch a validator is on the active committee. Year-1 yields are high while the validator pool is small and inflation is at the 5 % rate; the rate compresses as the pool grows and inflation tapers to the 1 % terminal floor.

13.5 Net Inflation

Net inflation = mint − burn. At sustained moderate usage (with realistic fee loads), the annual burn exceeds annual mint within a few years; the chain becomes net deflationary. At low usage, slight inflation maintains the validator security budget. At very high usage, deflationary pressure may eventually require parameter adjustment via governance.

13.6 Token Demand Drivers

PYDE has multiple protocol-internal demand drivers, not solely secondary-market speculation: validators stake PYDE to be eligible for committee selection (the active 128 are uniform-randomly selected from the eligible pool each epoch); parachain operators stake PYDE to run third-party parachains and earn the parachain's fees; every on-chain transaction (transfer, contract call, deploy, governance) pays base fees in PYDE; the treasury operates in PYDE. The fee structure (70% burned, 30% accrued to the validator reward pool + treasury) couples token value to chain usage rather than to trading volume — every base-fee unit either reduces supply or accrues to participants who secure or govern the network. For investors, the design intent is that long-term token utility tracks the chain's usefulness as infrastructure, not its position in a market cycle.

---

14. Governance

Pyde's governance is **off-chain**. Protocol changes proceed via Pyde Improvement Proposals (PIPs) — public, versioned, ratified by social consensus, modeled on Bitcoin's BIPs and Ethereum's EIPs. Validators upgrade voluntarily; hard forks happen by social agreement; the chain that retains 67 % + stake is the legitimate continuation.

On-chain governance is restricted to two surfaces: treasury spending and emergency operations, both gated by an M-of-N FALCON multisig (7-of-12 recommended) with a 30-day-bounded emergency-pause primitive.

Two-chamber on-chain governance was evaluated and explicitly rejected. Protocol upgrade should require coordinated human decision, not stake-weighted voting that incumbents can capture. The pattern of governance attacks across stake-weighted systems is the empirical case against this design. For institutional adopters, this means protocol evolution proceeds through public coordination — not silent stake-weighted votes that incumbents can capture — and the on-chain governance surface (treasury, emergency pause) is bounded, auditable, and recoverable.

---

15. Slashing

Pyde's slashing magnitudes are industry-aligned, with correlated-offense multipliers and an evidence-submitter reward for safety violations:

Offense	First instance	Max (correlation / repeat)
Equivocation	10 %	50 %
Bad state-root signature	10 %	50 %
Bad anchor attestation	5 %	20 %
Invalid vertex	5 %	30 %
Bad decryption share	5 %	30 %
DKG failure	2 %	10 %
Share withholding (per round)	0.1 %	5 % / epoch
Extended downtime (per round)	0.05 %	10 % / epoch
Bad batch attestation	2 %	5 %

Coordinated safety offenses apply a 2 × multiplier. Reporter receives 10 % of safety-slash distributions; the remainder is burned. A 24-hour slashing escrow window allows governance to void false positives. Jail escalation runs 24 h → 7 d → permanent on repeat liveness offenses.

---

16. Comparison to Other L1s

16.1 Comparison Matrix

Axis	Pyde	Ethereum (L1)	Solana	Aptos	Sui	Polkadot	Cosmos	Avalanche
Post-quantum signatures (default)	**Yes** (FALCON-512)	Roadmap	Roadmap	Roadmap	Roadmap	Roadmap	Roadmap	Roadmap
Encrypted mempool (default)	**Yes** (Kyber-768 threshold)	No (PBS auction)	No (Jito auction)	No	No	No	Proposals in IBC track	No
Sandwich-attack prevention	**Structural**	Partial (PBS)	Partial (Jito)	Partial	Partial	N/A (relay-chain)	Partial	Partial
Hard-finality time	~ 500 ms (DAG commit)	~ 12 min	Probabilistic (~ 13 s)	< 1 s	< 1 s	~ 12 – 60 s	~ 6 s	~ 1 s
Validator hardware	8c / 16 GB / 500 GB / 100 Mbps (awaiting committee)	Modest	12 + cores / 256 + GB	Modest	Modest	Modest (validator tier)	Modest (per zone)	Modest
Equal validator voting	**Yes** (1 = 1)	Stake-weighted	Stake-weighted	Stake-weighted	Stake-weighted	Stake-weighted	Stake-weighted	Stake-weighted
Permissionless cross-chain infra layer	Roadmap (parachain spec, PYDE-staked, unified gas)	L2s (per-L2 sequencer); third-party oracles	Third-party (Pyth / Switchboard)	No	No	App-chains via auctions	IBC zone-to-zone (no integrated infra)	Subnet model (sovereign, not infra)

Each chain in this matrix is competently engineered by serious teams. The differences are choices, not capability gaps. The matrix exists to make the choices visible, not to imply a ranking.

16.2 What Pyde Owes the Field

Pyde does not invent every wheel. The chain stands on a foundation the rest of the industry built — and the strategic claim is not that other chains are wrong, but that the time has come to integrate the field's best ideas into a single greenfield design.

- **Bitcoin** invented the field. Public chain, hard rules, minimal trust assumptions — the social model everything in this document presupposes.
- **Ethereum** invented the programmable blockchain and shaped most of the design vocabulary the field still uses: smart contracts, EVM execution semantics, the EIP process, EIP-1559, account-abstraction roadmap, the entire MEV literature. Pyde adopts the EIP-1559 base-fee + elastic-block design (with the priority-tip removal the encrypted mempool enables) and the EIP-style off-chain governance workflow.
- **Solana** proved at scale that a monolithic-binary L1 with parallel execution can deliver retail throughput, and that consensus and execution sharing one process is operationally viable. Pyde's monolithic architecture, access-list-driven scheduler, and sub-second-finality commitment are the same family of design choices Solana legitimized in

production. Solana's stability work — mempool overload mitigations, consensus liveness fixes, gossipsub tuning — is the production reference for what hardening a high-throughput chain looks like.

- **Aptos** contributed the Jellyfish Merkle Tree that Pyde adopts as its state structure, and Block-STM as one of the field's two leading approaches to the parallelism problem. Pyde's hybrid scheduler folds Block-STM in alongside static access lists.
- **Sui** introduced the object-centric model as one of the cleanest expressions of ownership encoded in the transaction structure. Pyde's scheduler operates against declared access lists rather than encoded ownership, but Sui's work established that parallelism is a function of transaction format, not just scheduler implementation. Mysticeti — the consensus Pyde adopts post-pivot — was developed by the Mysten Labs team behind Sui.
- **Narwhal and Bullshark** (Sonnino, Spiegelman, et al.) established the worker / primary split and the DAG-based mempool design Pyde's consensus directly builds on.
- **Polkadot** pioneered pluggable consensus (BABE / GRANDPA) and parachain architecture as a first-class concept. Pyde's parachain layer applies pluggable-consensus thinking to a different scope — decentralized infrastructure providers, not sovereign app-chains.
- **Cosmos** built IBC, the most rigorous cross-chain protocol shipped to date, and the principle that cross-chain interaction should be cryptographically verifiable rather than custodially trusted. Pyde's `HardFinalityCert`-based bridge primitive sits in IBC's intellectual lineage.
- **Avalanche** demonstrated that subnet-style horizontal scaling is operationally tractable and that Snowman / Avalanche consensus can deliver sub-second finality at production scale.
- **Chainlink** built the production reference for decentralized oracle networks — the operator-set staking model, deviation-tolerance attestations, off-chain-to-on-chain data bridges. Pyde's parachain layer is Chainlink-style decentralized infrastructure integrated natively into an L1's gas model.
- **Filecoin and the libp2p / IPFS ecosystem** produced the modular networking stack Pyde uses as transport. Pyde's net-layer crate is integration work over a stack the field built.
- **Cardano, Tezos, Algorand, Mina, Aleo, Diem, the Move language team, the entire ZK-rollup research community, Flashbots, the Rust async-runtime ecosystem, NIST, IETF** — all shaped the design space. The list is not exhaustive.

Where Pyde diverges — post-quantum-from-genesis, encrypted-mempool-by-default, equal voting, commodity hardware, the permissionless parachain layer with unified gas — is where the bet sits. Every chain in the comparison was built for the era it was built for. **Pyde is the only chain in the table that needs no migration to ship all four properties.**

For investors and adoption partners evaluating which L1 to build on, the practical question is not which chain is fastest in a benchmark but which chain's properties will still match the application's needs in 2030 and 2035. Pyde's bet is that the answer is the chain that started with those properties — quantum-resistant, MEV-resistant at the protocol layer, sub-second-final, commodity-validated, permissionlessly extensible — rather than the chain that has the longest migration to do them.

---

17. Open Problems

The design is complete in the senses that matter; the engineering risk is concentrated in a few specific places that this document calls out explicitly rather than hides.

17.1 Threshold Post-Quantum Cryptography

Production-grade threshold variants of Kyber are research-stage. Pyde v1 may ship with a classical-crypto threshold scheme (ElGamal-style over a high-prime field, used only inside the threshold-decryption ceremony) as a transitional measure, migrating to threshold Kyber when audited implementations mature. This is the single largest cryptographic engineering risk in the design. It is being actively researched, not skipped.

17.2 Batch Threshold Decryption

Per-ciphertext threshold decryption scales poorly at very high encrypted throughput — beyond a certain point the per-ceremony cost dominates on commodity hardware. Batch decryption schemes — where one threshold ceremony decrypts multiple ciphertexts amortized — are research-stage. Pyde v2 will adopt one once standardization matures; v1 ships the per-ceremony scheme with GPU-acceleration headroom characterized by the performance harness.

17.3 ZK Light Clients

The hybrid-hashing strategy (Poseidon2 on ZK-bearing paths) keeps zero-knowledge proof options open. Post-mainnet, ZK-validated state proofs would enable succinct light clients (kilobytes of proof, full security). Specific SNARK system choice (Plonky3, SP1, Halo2, RISC Zero) is deferred until the consensus rebuild lands and the per-circuit cost can be measured against real protocol structures.

17.4 Programmable Accounts and Session Keys

Native multisig ships at v1. **Programmable accounts** (sandboxed WASM policy modules expressing spend limits, time locks, allow-listed recipients, tiered authorization, recovery flows) and **session keys** (epoch-bounded, scope-limited dApp delegation without per-action wallet popups) ship post-mainnet.

A session key is bounded by four parameters: an allow-list of contracts, an optional allow-list of method selectors, a hard spend cap, and an expiry wave. Each authorization checks the FALCON signature, the liveness flags, the scope, and the cumulative spend — all four must pass. Revocation is a single tx signed by the account's main `auth_keys` and takes effect at the next wave commit. Ethereum is retrofitting the same idea via ERC-4337; Pyde gets it at the protocol layer.

The `AuthKeys` enum reserves the `Programmable` variant (tag `0x03`) at genesis. The account `code_hash` + `storage_root` shape and the multisig signature pipeline are also v1 surfaces that v2 reuses, so contracts written today survive the upgrade without rewriting. The full mechanism is documented in Chapter 11 *Session keys (v2)* and companion/DESIGN.md.

17.5 Parachain Layer

The protocol-level cross-chain primitives (`cross_call!`, `HardFinalityCert`) ship at genesis with mainnet stubs. The full parachain layer — specification, reference implementations, operator economics, bridges to Ethereum / Cosmos / Solana — ships post-mainnet.

---

18. Path to Mainnet

This document is the technical specification of the post-pivot design. The engineering between specification and mainnet is the work ahead, in execution order:

1. **Mysticeti DAG implementation.** Adapt the open-source Mysticeti reference for FALCON-bound signatures and Pyde's threshold-decryption integration; rebuild the consensus, mempool, and node crates against the new foundation.
2. **Performance harness build-out.** Multi-region production-realistic infrastructure; workload generators for the four target tx-mixes; chaos / failure injection; soak schedule. Pre-mainnet test slate is mandatory before any external TPS claim.
3. **External audit programme.** Multi-track, specialist firms across consensus, the WASM execution layer integration (host-function ABI, fuel-to-gas mapping, deploy-time validator), post-quantum cryptography, networking, and the `otigen` developer toolchain. Remediate all critical and high findings; re-audit the remediation. The wasmtime runtime itself is a vetted production dependency from the Bytecode Alliance and is not separately audited.
4. **Incentivized testnet.** Reference dApps (DEX, lending market, NFT marketplace); fully-funded bug bounty at mainnet-tier scale; multi-month soak; remediate community-found issues before launch.

5. **128-validator genesis.** Recruit operators with documented hardware benchmarks and incentivized-testnet participation. Geo-distribute across 3 + regions. Coordinate validator DKG for the threshold pubkey. Sign the genesis block. Publish the chain hash.

There is no public schedule. Mainnet ships when the audit programme passes and the incentivized testnet validates the throughput target on production-realistic infrastructure — not before. For investors, the absence of a public schedule is by design: the project prioritizes correctness over a date, and each milestone above is a gate that must close on the merits before mainnet.

---

19. Conclusion

Pyde represents a chain built around the architectural requirements of the next decade: post-quantum security, MEV resistance, sub-second finality, and commodity-hardware decentralization for users and infrastructure. The pivot from in-house HotStuff to Mysticeti-style DAG consensus reflects an explicit commitment to designing from a clean foundation rather than patching accumulated technical debt.

The design is complete; the implementation is the work ahead. This is not a chain that ships in six months. It is a chain that aims to occupy a category — post-quantum, MEV-resistant, commodity-validated — that no production chain occupies cleanly today. The strategic window for that occupancy is open and time-bound.

For businesses, that category means settlement infrastructure that holds through the next cryptographic generation, without a hidden tax on customer transactions and without a coordinated migration to budget for. For developer teams, it means a permissionless surface to launch their own execution environments and bootstrap dev communities around them, on top of a runtime they already know how to write. For users, it means trades that execute at the price signed and funds that remain valid through whatever the next decade brings cryptographically. The architecture is the bet; the implementation work is what turns the bet into a chain people can actually use.

Document version: 0.2 **Status:** Living document **License:** Apache-2.0 — see `LICENSE` at the repository root