WitnessChain: Preprint
Pre-Genesis Trust Anchoring, Threshold Witness Governance, and Sealed Computation Surfaces

Preprint v 0.2

Mayckon Giovani   WitnessChain Research Collective
contact@stigning.com

March 2026 — *Preprint (condensed; revision 2026-03-20)*

Abstract

WitnessChain is a sovereign Layer-1 protocol whose trust root is not a single genesis artifact but a *pre-genesis record*: a commitment to the future genesis state co-signed by a hardware-bound witness set under a high-threshold signature scheme. The whitepaper positions this primitive as a resolution to the *Genesis Problem* (undetectable manipulation of initial state), and as the foundation for post-quantum security, threshold governance, and sealed computation surfaces.

This preprint condenses the full LaTeX technical report into: (i) a formal system model and adversary model, (ii) the pre-genesis primitive as a verifiable commitment and threshold ceremony, (iii) the threshold Dilithium DKG surface (including the distributed-abort constraint), (iv) the deterministic parallel execution model via WorkPackages and State Access Lists (SAL), (v) the privacy architecture (ZK, zkVM, and sealed computation via FHE), and (vi) the tamper-aware data availability (DA) state machine with erasure-coded recovery.

Keywords: pre-genesis anchoring; threshold signatures; post-quantum cryptography; TEEs; zk-SNARKs; FHE; AURA; GRANDPA; VDF; data availability.

Contents

1 Problem Statement (Condensed)

The whitepaper motivates WitnessChain by asserting four structural deficiencies in existing public blockchains: (i) the genesis block is a non-verifiable trust root; (ii) base-layer transparency makes privacy an overlay rather than an invariant; (iii) classical signature schemes create a *quantum cliff* and enable harvest-now/decrypt-later attacks; and (iv) plaintext transaction visibility enables extractable value via ordering manipulation.

2 System Model and Notation

Actors. Let $\mathcal{W} = \{w_1, \ldots, w_n\}$ be the witness registry with $n = 101$. Witnesses are assumed to operate within Trusted Execution Environments (TEEs) such as Intel SGX / ARM TrustZone, and participate in both consensus and threshold authorization. An *Owner* O proposes the pre-genesis record.

Cryptographic Interfaces. Let $\mathcal{H}$ be a collision-resistant hash function. Let $(\mathsf{Sign}, \mathsf{Verify})$ be a post-quantum signature scheme (the whitepaper names CRYSTALS-Dilithium/Falcon). We write $\Sigma_t \leftarrow \mathsf{Combine}(\{\sigma_i\}_{i\in S})$ for a threshold signature derived from $|S| \geq t$ partial signatures $\sigma_i \leftarrow \mathsf{PartSign}_{sk_i}(m)$.

Thresholds. The whitepaper specifies multi-threshold authorization with two salient levels:

- **Routine threshold** $(6, 101)$ for operational actions (e.g., restricted privacy access, token metadata operations).
- **Sensitive threshold** $(90, 101)$ for safety-critical actions (e.g., absolute privacy access, tamper recovery, hard forks).

3 Threat Model and Security Assumptions

We model the adversary $\mathcal{A}$ as a probabilistic polynomial-time (PPT) algorithm that may additionally be quantum-capable (denoted $\mathcal{A}^Q$). In the whitepaper, $\mathcal{A}$ may:

- **Statically corrupt** up to $f \leq 33$ witnesses before protocol execution (i.e., fewer than 1/3 of a 101-witness committee).
- **Adaptively corrupt** up to $f' < 12$ witnesses during execution for sensitive operations (notably $(90, 101)$ authorization).
- **Control the network** (deliver, delay, or reorder messages) within a bounded delay Δ, but not suppress messages indefinitely.
- **Observe** all on-chain state, ciphertexts, and proofs.

The adversary cannot: (i) break MLWE/MSIS hardness at the chosen Dilithium parameters, (ii) break IND-CPA of the selected FHE scheme, (iii) observe the interior of honest TEE enclaves, or (iv) invert the VDF faster than sequential evaluation.

3.1 Trust Hierarchy

The protocol's assumptions form an explicit trust hierarchy:

$$\underbrace{\text{Hardware Fabricant}}_{\text{TEE root of trust}} \succ \underbrace{\text{Witness TEE}}_{\text{key isolation}} \succ \underbrace{\text{Witness supermajority}}_{\geq 90 \text{ of } 101} \succ \underbrace{\text{Protocol}}_{\text{on-chain rules}} \succ \underbrace{\text{User}}_{\text{no trust required}}$$

3.2 Safety and Liveness

Definition 3.1 (Safety)**.** The protocol satisfies *safety* if no two honest nodes ever accept conflicting finalized blocks $B \neq B'$ at the same height.

Definition 3.2 (Liveness)**.** The protocol satisfies *liveness* if every transaction submitted by an honest user is eventually included in a finalized block.

Proposition 3.3 (Safety under GRANDPA)**.** The GRANDPA finality gadget guarantees safety when fewer than 1/3 of weighted votes are Byzantine. Under the stated corruption bounds and the staking model, the Byzantine fraction remains $< 1/3$ with overwhelming probability.

4 Cryptographic Preliminaries (Minimal)

This preprint treats primitive implementations as interchangeable *interfaces* but is explicit about which assumptions sit at each trust boundary.

4.1 Threshold Signatures (Interface)

Definition 4.1 (Threshold Signature Interface)**.** Let $t \leq n$. A (t, n) threshold signature scheme exposes algorithms:

$$\mathsf{KeyGen},\ \mathsf{Share},\ \mathsf{PartSign},\ \mathsf{Combine},\ \mathsf{Verify},$$

such that any set S with $|S| \geq t$ can produce a signature $\Sigma \leftarrow \mathsf{Combine}(\{\sigma_i\}_{i \in S})$ that verifies under the master public key, while any coalition of $< t$ parties cannot forge signatures except with probability $\mathsf{negl}(\lambda)$.

Remark 4.2 (BLS vs. T-Dilithium)**.** For BLS, $\mathsf{Combine}$ is a linear combination in a pairing group. For Threshold Dilithium, $\mathsf{Combine}$ is a multi-round Fiat–Shamir transcript with rejection sampling, which introduces the *distributed abort* constraint discussed in §6. The notation above abstracts these details while keeping the security boundary explicit.

4.2 Zero-Knowledge Proof Systems

Definition 4.3 (zk-SNARK Interface)**.** Let $\mathcal{R} = \{(x, w) : C(x, w) = 1\}$ be an NP relation. A zk-SNARK exposes $\mathsf{Prove}(\mathsf{crs}, x, w) \to \pi$ and $\mathsf{Verify}(\mathsf{crs}, x, \pi) \to \{0, 1\}$ and satisfies completeness, knowledge soundness, and zero-knowledge.

4.3 FHE (Sealed Computation)

Definition 4.4 (FHE Interface)**.** A fully homomorphic encryption scheme exposes $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ such that for any function f in the supported class:

$$\mathsf{Dec}(sk, \mathsf{Eval}(f, \mathsf{Enc}(pk, m))) = f(m),$$

and ciphertexts are IND-CPA secure under the underlying hardness assumption (e.g., Ring-LWE for TFHE-style constructions).

4.4 VDF (Unbiasable Randomness)

Definition 4.5 (VDF Interface)**.** A Verifiable Delay Function exposes $(\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ where Eval requires $\Theta(T)$ sequential work and Verify is efficient. Sequentiality prevents precomputation and grinding of the output for a fixed challenge input.

5 Pre-Genesis Block Construction

We define the pre-genesis anchoring protocol Π_{PG} as a tuple of polynomial-time algorithms binding a planned genesis state to a threshold witness signature.

5.1 Formal Construction (Condensed)

Let $n = 101$ and threshold $t = 90$. Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^{256}$ be a collision-resistant hash and let $(\mathsf{Sign}, \mathsf{Verify})$ denote post-quantum signing interfaces instantiated by Threshold Dilithium.

PreGenesisSetup(1^λ)**:** Sample Dilithium parameters at NIST Level 3 and generate a $(90, 101)$-DKG session producing a master public key mpk and per-witness secret shares $(sk_1, \dots, sk_{101})$. Output public parameters pp.

WitnessRegister$(\mathsf{pp}, w_i, \mathsf{id}_i)$**:** Each witness presents a hardware binding attestation att_i from a TEE proving knowledge of sk_i without revealing it, producing a registry record $\mathsf{reg}_i = (\mathsf{id}_i, pk_i, \mathsf{att}_i, \mathsf{addr}_i)$. Publish $\mathcal{W} = \{\mathsf{reg}_1, \dots, \mathsf{reg}_{101}\}$.

CommitGen$(O, G_{\mathsf{planned}})$**:** Construct planned genesis state G_{planned} and compute:

$$H_{\mathsf{fut}} = \mathcal{H}(G_{\mathsf{planned}}), \qquad q_{\mathsf{pre}} = \sum_j \mathsf{allocation}(a_j), \qquad T_{\mathsf{ts}} = \mathsf{UnixTime}().$$

Broadcast $m_{\mathsf{commit}} = H_{\mathsf{fut}} \| \mathcal{H}(\mathcal{W}) \| q_{\mathsf{pre}} \| T_{\mathsf{ts}}$.

ThresholdSign$(\mathsf{pp}, S, m_{\mathsf{commit}})$**:** Each witness $w_i \in S$ (with $|S| \geq t$) verifies the commitment and outputs a partial signature $\sigma_i \leftarrow \mathsf{PartSign}_{sk_i}(m_{\mathsf{commit}})$. The combiner computes the aggregated threshold signature $\Sigma_{\mathsf{PQC}} \leftarrow \mathsf{Combine}(\{\sigma_i\}_{i \in S})$.

GenesisActivate$(\mathsf{pp}, G_0, \mathsf{PG})$**:** Publish $G_0 = G_{\mathsf{planned}}$ and the pre-genesis record

$$\mathsf{PG} = (H_{\mathsf{fut}},\ \mathcal{W},\ q_{\mathsf{pre}},\ T_{\mathsf{ts}},\ \Sigma_{\mathsf{PQC}}). \tag{1}$$

Verify$(\mathsf{pp}, \mathsf{PG}, G_0)$**:** Check (i) $\mathcal{H}(G_0) = H_{\mathsf{fut}}$, (ii) $\mathsf{Verify}(\mathsf{mpk}, m_{\mathsf{commit}}, \Sigma_{\mathsf{PQC}}) = 1$, and (iii) $T_{\mathsf{ts}} \leq T_{\mathsf{now}}$.

Algorithm 1 Pre-Genesis Anchoring Protocol Π_{PG} (condensed)

```
 1: $\mathsf{pp} \leftarrow \textsc{PreGenesisSetup}(1^\lambda)$
 2: for $i \leftarrow 1$ to $101$ do
 3:     $\mathsf{reg}_i \leftarrow \textsc{WitnessRegister}(\mathsf{pp}, w_i, \mathsf{id}_i)$
 4: end for
 5: $(H_{\mathsf{fut}}, q_{\mathsf{pre}}, T_{\mathsf{ts}}) \leftarrow \textsc{CommitGen}(O, G_{\mathsf{planned}})$
 6: Choose $S \subseteq \mathcal{W}$ with $|S| \geq 90$
 7: $\Sigma_{\mathsf{PQC}} \leftarrow \textsc{ThresholdSign}(\mathsf{pp}, S, m_{\mathsf{commit}})$
 8: $\mathsf{PG} \leftarrow (H_{\mathsf{fut}}, \mathcal{W}, q_{\mathsf{pre}}, T_{\mathsf{ts}}, \Sigma_{\mathsf{PQC}})$
 9: Publish $\mathsf{PG}$ and $G_0 = G_{\mathsf{planned}}$
10: assert $\textsc{Verify}(\mathsf{pp}, \mathsf{PG}, G_0) = 1$
```

5.2 Security Properties (Statements)

Definition 5.1 (Pre-Genesis Integrity)**.** Protocol Π_{PG} satisfies *Pre-Genesis Integrity* if for all PPT adversaries $\mathcal{A}$ controlling $c < 12$ witnesses:

$$\Pr[\textsc{Verify}(\mathsf{pp}, \mathsf{PG}^*, G_0^*) = 1 \ \wedge\ \mathcal{H}(G_0^*) \neq H_{\mathsf{fut}}] \leq \mathsf{negl}(\lambda). \tag{2}$$

Definition 5.2 (Witness Accountability)**.** If two valid pre-genesis records $\mathsf{PG}_0, \mathsf{PG}_1$ with $H^{(0)}_{\mathsf{fut}} \neq H^{(1)}_{\mathsf{fut}}$ are published, an extractor identifies a set $\mathcal{C} \subseteq [101]$ of witnesses that signed both, with

$$|\mathcal{C}| \geq 90 + 90 - 101 = 79, \tag{3}$$

except with probability $\mathsf{negl}(\lambda)$.

Definition 5.3 (Genesis Finality)**.** The protocol satisfies *Genesis Finality* if the threshold committee can commit to at most one future genesis state (up to $\mathsf{negl}(\lambda)$), and any equivocation is attributable via Witness Accountability.

Definition 5.4 (Genesis Validity)**.** A genesis block $\mathsf{Genesis}$ is valid with respect to PG iff

$$\mathcal{H}(\mathsf{Genesis}) = H_{\mathsf{fut}} \;\wedge\; \mathsf{Verify}_{\mathsf{PQC}}(\Sigma,\ \mathsf{PG},\ \mathcal{W} \cup \{O\}) = 1. \tag{4}$$

Theorem 5.5 (Pre-Genesis Unforgeability (Informal))**.** Under lattice hardness assumptions (MLWE/MSIS) and in the random oracle model, no PPT adversary controlling fewer than $101 - t$ witnesses can produce a valid forgery for a conflicting future genesis hash, except with probability $\mathsf{negl}(\lambda)$.

6 Threshold Dilithium (T-Dilithium) DKG Summary

All threshold signing is instantiated over $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ with $q = 8{,}380{,}417$ and $n = 256$ (Dilithium3-level parameters). A public matrix $\mathbf{A} \in R_q^{k \times \ell}$ is derived from a public seed ρ, ensuring no party controls $\mathbf{A}$.

6.1 Distributed Abort Constraint (Why Threshold Dilithium is Non-Trivial)

Unlike pairing-based aggregation, Dilithium signatures rely on the Fiat–Shamir-with-aborts paradigm: a witness must reject and resample if its response violates a norm bound. In a threshold setting, naively restarting the entire committee on any abort yields catastrophic latency.

Definition 6.1 (Distributed Abort Event)**.** Let p_{abort} be the per-witness abort probability induced by the Dilithium norm test. For an active signing set S with $|S| = t$, the probability that *at least one* witness aborts in a single synchronous round is:

$$p_{\mathsf{group}} = 1 - (1 - p_{\mathsf{abort}})^t. \tag{5}$$

For Dilithium3-like parameters, p_{abort} is non-negligible; thus $p_{\mathsf{group}} \approx 1$ for $t = 90$ unless aborts are handled independently.

Proposition 6.2 (Completion via Replacement)**.** If the active set is drawn from an extended committee of size $m > t$ and aborting witnesses are replaced without restarting already-valid responses, the expected number of attempts to collect t valid partials is:

$$\mathbb{E}[\text{tries}] = \frac{t}{1 - p_{\mathsf{abort}}}. \tag{6}$$

This is linear rather than exponential in t.

6.2 Distributed Key Generation (DKG)

Each witness P_i samples local secrets $(\mathbf{s}_{1,i}, \mathbf{s}_{2,i})$ and computes a local public contribution:

$$\mathbf{t}_i = \mathbf{A}\mathbf{s}_{1,i} + \mathbf{s}_{2,i} \in R_q^k.$$

Witnesses execute verifiable secret sharing (VSS) over R_q^ℓ by sharing $\mathbf{s}_{1,i}$ via degree-$(t-1)$ polynomials and broadcasting commitments. Each witness P_j verifies received shares by checking the committed relation under $\mathbf{A}$. The long-term secret share held by P_j is the sum of verified shares, and the master public key is:

$$\mathbf{t}^{\mathsf{master}} = \sum_{i=1}^{101} \mathbf{t}_i = \mathbf{A}\left(\sum_{i=1}^{101} \mathbf{s}_{1,i}\right) + \sum_{i=1}^{101} \mathbf{s}_{2,i}. \tag{7}$$

6.3 Threshold Signing Interface

For a message M, each witness produces a partial signature using its share; aggregation uses a multi-round transcript and share-combining coefficients over $\mathbb{Z}_q$ without reconstructing any master secret in the clear. The full paper provides a QROM SUF-CMA security argument via measure-and-reprogram; this preprint treats that result as an assumption surface and focuses on the trust boundaries and invariants.

7 Consensus and Finality (AURA + GRANDPA + VDF)

The whitepaper specifies block production via AURA (slot-based leader rotation) and finality via GRANDPA (supermajority voting). Let slot index s select leader $w_{s \bmod n}$. GRANDPA finalizes blocks when $\lceil 2n/3 \rceil$ votes are observed. Randomness is seeded by a Verifiable Delay Function (VDF) beacon to reduce leader-bias in witness selection.

7.1 VDF Randomness (Grinding Resistance)

Let $(\mathsf{Setup}, \mathsf{Eval}, \mathsf{Verify})$ be the VDF interface. Slot randomness for epoch e is derived as:

$$r_e = \mathsf{Eval}(\mathcal{H}(\mathsf{epoch\_seed}_e),\, T_{\mathsf{slot}}), \tag{8}$$

where T_{slot} is chosen so Eval completes at the slot boundary. Sequentiality implies a witness cannot evaluate multiple candidate seeds in parallel to bias leader selection.

8 Execution Model: WorkPackages and State Access Lists

WitnessChain targets deterministic parallel execution by forcing *explicit* state boundaries. The basic unit of execution is a *WorkPackage* with a declared access mask. Scheduling is therefore a pure function of declared access sets, not runtime discovery.

8.1 WorkPackages (Condensed)

A WorkPackage p contains a payload (call data), a code hash identifying the execution circuit/program, and a bitmask `disjoint_access_list` representing state shards the package may touch. The proposer partitions the block into batches such that, for any two packages executed in parallel, their masks are disjoint.

Proposition 8.1 (Parallel Determinism under Disjoint Access)**.** If p_i and p_j are scheduled in the same parallel batch and

$$p_i.\texttt{disjoint\_access\_list} \,\&\, p_j.\texttt{disjoint\_access\_list} = \mathbf{0},$$

then the resulting state mutations commute; the merged state root is unique and independent of thread scheduling.

8.2 State Access Lists (SAL)

The design problem: for non-trivial programs (e.g., DEX routers), the set of state shards touched is often only known after evaluating some state-dependent branch. The whitepaper frames the proposer trilemma as:

1. **Pessimistic declaration**: declare too much state, collapsing parallelism.
2. **Optimistic under-declaration**: discover a conflict at merge time, breaking determinism.
3. **Sender declaration**: require the sender to declare access up-front; abort on undeclared access.

The protocol adopts option 3 as a first-class invariant.

Definition 8.2 (State Access List)**.** A *State Access List* (SAL) for transaction tx is a tuple

$$\mathsf{SAL}(\mathsf{tx}) = (\mathcal{S}_{\mathsf{slots}}, \mathcal{C}_{\mathsf{hashes}}, \mathcal{K}_{\mathsf{fhe}}),$$

where $\mathcal{S}_{\mathsf{slots}}$ are storage slot/shard indices tx may read/write, $\mathcal{C}_{\mathsf{hashes}}$ are contract/circuit code hashes it may invoke, and $\mathcal{K}_{\mathsf{fhe}}$ are FHE key handles it may use for sealed computation. The SAL is co-signed with the transaction authorization proof; forging it is equivalent to breaking authentication.

Definition 8.3 (Deterministic Abort on SAL Violation)**.** An *SAL violation* occurs if execution attempts to read/write $s \notin \mathcal{S}_{\mathsf{slots}}$ or call $h \notin \mathcal{C}_{\mathsf{hashes}}$. On violation, the protocol performs a *Deterministic Abort*:

1. Discard all state mutations from tx atomically (no partial writes).
2. Charge only a base fee f_{base} (not resource fees).
3. Record the abort as a typed receipt event; the block remains valid.

Algorithm 2 SAL Guarded Execution (condensed)

```
Require: Transaction $\mathsf{tx}$ with SAL $(\mathcal{S}_{\mathsf{slots}}, \mathcal{C}_{\mathsf{hashes}}, \mathcal{K}_{\mathsf{fhe}})$
1: Initialize empty changeset $\Delta$
2: for each state access $(x, \mathsf{op})$ during execution do
3:     if $x \notin \mathcal{S}_{\mathsf{slots}}$ then
4:         abort: discard $\Delta$; charge $f_{\mathsf{base}}$; emit receipt
5:     end if
6:     charge gas for $\mathsf{op}$; apply access to $\Delta$
7: end for
8: commit $\Delta$ to the Merkle state
```

Theorem 8.4 (Deterministic Abort Safety)**.** Under the SAL model above, block execution satisfies simultaneously:

1. **State determinism:** SAL-violating transactions produce zero state changes.
2. **Block validity:** violations are typed receipts, not invalid blocks.
3. **Parallelism soundness:** disjoint masks derived from the *declared* SAL remain conservative; any undeclared access triggers abort.

Proposition 8.5 (DEX Concurrency Under SAL)**.** For a multi-hop DEX router touching pools in shards $\{s_1, \ldots, s_h\}$:

1. If the sender declares $\{s_1, \ldots, s_h\}$, parallelism matches the ideal disjoint-shard case.
2. If the sender pessimistically declares all pool shards, parallelism collapses but the access pattern becomes auditable.
3. If the sender under-declares, a Deterministic Abort occurs and the sender re-submits with the corrected SAL; no block invalidity is induced.

8.3 Join-Accumulate Accumulator (JAM Surface)

SAL provides the *declared* boundaries. The accumulator provides the merge operator.

Definition 8.6 (Accumulation)**.** Let σ be the global state and let $\{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$ be a set of WorkPackages whose SAL masks are pairwise disjoint. Accumulation is defined as:

$$\sigma' = \sigma \oplus \bigoplus_{i=1}^{k} \mathsf{Apply}(\sigma_i, \mathcal{P}_i), \tag{9}$$

where $\oplus$ is a conflict-free merge over disjoint state segments.

Invariant 8.7 (No Mutex, No Hidden State)**.** Any state read/write performed by execution must be accounted for by the SAL. There is no implicit shared memory: conflicts are defined only over explicit state keys. Therefore, the scheduler is a deterministic function of the block payload.

9 zkVM and Recursive Proof Surfaces (Condensed)

WitnessChain treats "execution" as an object that can be proven. In the zkVM setting, a program P over a RISC-V-like ISA is wrapped by a proving system such that:

$$\pi_P \leftarrow \mathsf{Prove}(\mathsf{crs},\ x_{\mathsf{pub}},\ (P, x_{\mathsf{priv}})) \quad \text{s.t.} \quad P(x_{\mathsf{pub}}, x_{\mathsf{priv}}) = y. \tag{10}$$

Recursive aggregation produces constant-size proofs of many steps ("proof of proofs"), reducing verifier cost for light clients and bridges.

10 Privacy Architecture

10.1 Adaptive Privacy Tiers (Threshold-Gated Surfaces)

Definition 10.1 (Privacy Tier)**.** Let $\mathcal{T} \in \{\mathsf{Public}, \mathsf{Restricted}, \mathsf{Absolute}\}$. For an object X (block payload, ciphertext, or sealed state), define the access predicate:

$$\mathsf{Access}(X, \mathcal{T}) = \begin{cases} \top & \mathcal{T} = \mathsf{Public}, \\ \mathsf{Verify}^{(6,101)}(\Sigma_6, X) = 1 & \mathcal{T} = \mathsf{Restricted}, \\ \mathsf{Verify}^{(90,101)}(\Sigma_{90}, X) = 1 & \mathcal{T} = \mathsf{Absolute}. \end{cases} \tag{11}$$

10.2 Zero-Knowledge Transaction Layer (Statement)

Each transaction is verified by a circuit C_{tx} over a relation $\mathcal{R}_{\mathsf{tx}}$. The public statement includes commitments and nullifiers; the private witness includes spend keys and plaintext balances.

Theorem 10.2 (Transaction Privacy (Informal))**.** Under knowledge soundness and zero-knowledge of the zk-SNARK and PRF pseudorandomness for nullifier derivation, no PPT adversary can distinguish private transaction witnesses from simulated ones except with probability $\mathsf{negl}(\lambda)$.

10.3 Sealed Computation via FHE

Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be instantiated by TFHE-style encryption. Contracts operating on private state submit ciphertexts plus a ZK proof of correct encryption:

$$\pi_{\mathsf{fhe}} \leftarrow \mathsf{Prove}(\mathsf{crs},\ x = c,\ w = (m, r) \text{ s.t. } c = \mathsf{Enc}(pk, m; r)). \tag{12}$$

On-chain verification checks only (c, π_{fhe}); no plaintext state is required at any point in the validator pipeline.

10.4 Privacy Budget and Leakage Accounting

Nullifier Unlinkability. Each spent note produces a pseudorandom nullifier $\eta = \mathsf{PRF}_{sk}(\rho)$. The public nullifier set $\mathcal{N}$ is computationally indistinguishable from uniform under PRF security.

Stealth Address Budget. One-time recipient addresses take the form:

$$pk_{r,i} = r_i \cdot G + H_s(r_i \cdot pk_s) \cdot G, \tag{13}$$

with fresh $r_i \overset{\$}{\leftarrow} \mathbb{Z}_p$ per transaction. Observing k addresses for one recipient yields no more advantage than k random group elements under DDH/ROM assumptions.

ORAM Access Leakage. For N stored blocks, PathORAM leaks $O(\log N)$ bits per access (path length). Over q accesses:

$$\mathsf{Leak}_{\mathsf{ORAM}}(q) = O(q \log N) \text{ bits}. \tag{14}$$

11 Tamper-Aware Block Lifecycle and Data Availability

11.1 Tamper State Machine

Definition 11.1 (Block State Machine)**.** Let each block payload be associated to an access-attempt counter $\mathsf{cnt} \in \mathbb{N}$ and a state $\mathsf{state} \in \Omega$ where:

$$\Omega = \{\mathsf{Active}, \mathsf{Warned}, \mathsf{Tripped}, \mathsf{Shredding}, \mathsf{Burned}, \mathsf{Recovering}, \mathsf{Reminted}\}.$$

Transitions are triggered by cnt thresholds and by sensitive threshold approvals in the Burned state.

11.2 Erasure-Coded DA (RS$(200, 100)$)

When entering $\mathsf{Shredding}$, an encrypted payload $\mathsf{Enc}(pk, d)$ is encoded into $n_0 = 200$ shards with $k_0 = 100$ data shards:

$$\{\mathsf{shard}_j\}_{j=1}^{200} = \mathsf{RS.Enc}(\mathsf{Enc}(pk, d)). \tag{15}$$

Any k_0 shards reconstruct the payload.

11.3 Data Availability Sampling (DAS)

Light clients sample t shards uniformly at random. The probability that a withheld block passes t checks is bounded by:

$$\mathbb{P}\mathrm{r}[\text{unavailable} \mid \text{passes } t \text{ checks}] \leq \left(1 - \frac{k_0}{n_0}\right)^t = 2^{-t}. \tag{16}$$

For $t = 30$, the false-positive rate is $< 10^{-9}$.

Theorem 11.2 (DA Recovery Completeness)**.** If at least $k_0 = 100$ of the $n_0 = 200$ shards are available on the DA layer and the sensitive threshold $(90, 101)$ is achieved, the original payload is reconstructed with probability 1.

12 Hardware-Bound Witness Nodes (TEE Surface)

Witness signing shares and decryption shares are assumed to be sealed inside a TEE boundary. Remote attestation binds an enclave measurement to a witness public key.

Definition 12.1 (Remote Attestation Quote)**.** Witness w_i produces a quote:

$$Q_i = \mathsf{Sign}_{\mathsf{IK}}(\mathsf{MRENCLAVE} \| pk_i^{\mathsf{Dil}} \| T_{\mathsf{ts}}), \tag{17}$$

where IK is a hardware-backed attestation key and $\mathsf{MRENCLAVE}$ is the enclave measurement. The chain accepts w_i iff Q_i verifies against an allowlisted measurement set and freshness window.

Remark 12.2 (TEE Reality)**.** TEEs reduce the key-exfiltration surface but do not eliminate side-channel risk. The threat model explicitly excludes physical extraction and microarchitectural breaks of the vendor root-of-trust.

13 Post-Quantum Invariants and Ingress Hardening (Condensed)

The protocol uses post-quantum signatures from Block 0; the consensus and governance layer avoids a BLS dependency by instantiating threshold authorization under Dilithium-family assumptions. Ingress validation is treated as a DoS boundary: malformed lattice inputs must be rejected before they can trigger worker panics under parallel execution. The implementation surface uses SIMD-friendly checks for norm bounds and encoding validity.

14 Current Phase (Month 2 — Quantum Fortress)

This document describes the target invariant surface. Engineering work in Month 2 focuses on: (i) Threshold Dilithium DKG and signing transcript stabilisation (distributed abort handling; share sealing), (ii) SGX enclave integration for witness key custody and remote attestation, and (iii) MPC channels for committee coordination (complaints, replacement, and sensitive threshold actions).

15 Scope Notes

This preprint is intentionally condensed but not superficial. The source report expands full proofs, correlated-failure availability, cryptoeconomic analysis, and implementation detail.