Epic 1 – Enclave-Bound Anonymous Ingress & Mixer Network

Goal: Transactions enter the network completely blind, routed through hardware-attested TEEs

Story ID	User Story	Acceptance Criteria
ING-01	As a user, I can generate an enclave-routable onion transaction with 3–5 hops	- Transaction is < 4 KB serialized - Contains encrypted payload + 5-layer onion (public keys of TEE relays) - Signed with Dilithium key
ING-02	As a relay node, I can receive, decrypt one layer, attest in TEE, and forward without logging	- Remote attestation report included in every forward - Measurement hash = known good relay binary - No plaintext ever written to disk or untrusted RAM
ING-03	As the final relay, I can decrypt the inner payload and submit to shard ingress queue	- Payload is a ZK-wrapped transaction (Halo2 circuit) - Final relay adds its attestation + timestamp
ING-04	As a light client, I can discover 100+ active attested relays via DHT	- DHT returns only nodes with valid, unexpired attestation reports - Client randomly samples 5 relays per tx

ING-05	As the network, I enforce cover traffic & timing obfuscation	- Every relay injects chaff packets (configurable 1–10× real traffic) - Inter-packet delay jitter ±200 ms (AI-tuned)
ING-06	As an auditor, I can cryptographically prove no relay ever linked sender → tx	- Full simulation replay with all attestation reports passes unlinkability test (k=5 anonymity set > 10^6)

Epic 2 – Verifiable Delay Witnesses & Blind Validation

Goal: Any user can prove a tx is canonical with a ~1 KB witness, no full chain download

Story ID	User Story	Acceptance Criteria
BV-01	As a shard node, I can generate a 1–2 KB VDW for any committed tx	- Contains tx hash, shard ID, lattice height, embedding delta, TEE signature - Size ≤ 1800 bytes
BV-02	As a light client, I can verify a VDW in < 80 ms on iPhone 15	- Uses halo2 recursive verification - No network calls after witness receipt
BV-03	As a node, I serve VDWs over HTTP/3 with range-proof caching	- CDN-compatible, 5-year witness archival guarantee
BV-04	As a wallet, I can request and permanently cache VDWs for my txs	- Offline validation works forever

BV-05	As the protocol, I guarantee VDW non-repudiability even if shard reorgs	- Reorgs invalidate old VDWs and issue new ones automatically

Epic 3 – Neural State Embeddings & Homomorphic Updates

Goal: Replace Merkle trees with 512-byte AI embeddings that still allow ZK proofs

Story ID	User Story	Acceptance Criteria
EMB-01	Define and implement the base "LatentLedger" circuit (Halo2) that maps arbitrary state → 512-byte embedding	- Round-trip reconstruction error = 0 for balances up to 2^{256}
EMB-02	Implement homomorphic addition/subtraction on embeddings	- embed(A + Δ) = embed(A) + homomorphic_delta(Δ)
EMB-03	Train initial 24-layer transformer compressor on synthetic tx dataset	- Compression ratio ≥ 800× vs raw state - Model size ≤ 8 MB (fits in TEE)
EMB-04	Implement embedding inclusion proof (tx → embedding path)	- Proof size ≤ 800 bytes
EMB-05	Nodes periodically re-embed entire shard state inside TEE and publish new root	- Every 1000 txs or 10 s, whichever first

Epic 4 – AI-Native Optimistic Consensus

Goal: Sub-second finality via neural voting + cryptographic fallback

Story ID	User Story	Acceptance Criteria
CON-01	Nodes broadcast predicted post-tx embedding + BLS partial signature	- Prediction made with distilled on-device model
CON-02	Implement 67% embedding hash quorum → instant finality	- If 67% of weighted stake agree on hash → finalize
CON-03	Implement challenge phase with Monte-Carlo dispute resolution in TEE	- Losing side slashed 0.1–5% stake
CON-04	Implement reputation score updated via federated learning gradients	- Score influences voting weight (stake × reputation)
CON-05	Achieve median 600 ms probabilistic finality, 1.8 s cryptographic	- Measured on 500-node testnet with 5 continents

Epic 5 – Dynamic Neural Sharding

Goal: Auto-scale to 1000+ shards with zero manual config

Story ID	User Story	Acceptance Criteria
SHD-01	Implement shard load predictor (LSTM on tx rate, size, gas)	- Predicts overload 15 s in advance with >95% accuracy

SHD-02	Implement live shard split protocol (state embedding bisect)	- Split completes in < 4 s, no downtime
SHD-03	Implement shard merge when underutilized	- Merge threshold < 10 TPS sustained
SHD-04	Implement AI-driven erasure coding placement (5–7 replicas)	- Survives 40% node loss without data loss

Epic 6 – Useful-Work Economy & Federated Learning

Goal: Nodes earn by improving the collective AI

Story ID	User Story	Acceptance Criteria
UW-01	Nodes submit encrypted gradients after every 1000 txs	- Uses Secure Aggregation protocol
UW-02	Parameter server (run in TEE cluster) aggregates and publishes new model version	- Every 10 minutes
UW-03	Implement on-chain model registry with staking governance	- 7-day voting delay
UW-04	Pay gradient contributors proportional to model improvement (Shapley-value approximation)	- Paid in native token

Epic 7 – Quantum-Resistant Cryptography Suite

Goal: Day-1 post-quantum security

Story ID	User Story	Acceptance Criteria
QR-01	Replace all ECDSA/EdDSA with CRYSTALS-Dilithium (level 3)	- Keygen, sign, verify fully tested
QR-02	Implement ML-KEM (Kyber) for enclave-to-enclave key exchange	
QR-03	Implement SPHINCS+ as stateless backup for cold wallets	
QR-04	Implement migration path: old keys can co-exist for 2 years	

Epic 8 – Extreme Scalability Layer

Goal: 500k+ TPS on mainnet

Story ID	User Story	Acceptance Criteria
SCL-01	Parallel execution engine across 500 shards	- No cross-shard tx blocking
SCL-02	Native account abstraction & fee sponsorship	- Users pay zero gas if sponsored

SCL-03	AI fee predictor API (exact fee 10 s in advance)	- Accuracy > 99.9%
SCL-04	Recursive embedding compression → 900× smaller proofs than Polygon zkEVM	

Epic 9 – Developer Experience & SDK

Goal: Feels like writing a React app

Story ID	User Story	Acceptance Criteria
DX-01	Release Rust + TypeScript SDK with built-in ZK & embedding generation	
DX-02	“hln deploy” CLI that compiles, proves, and deploys private contracts in one command	
DX-03	One-click light client for React Native & Web (WebAssembly + TEE fallback)	
DX-04	AI auditor flags 95% of reentrancy, overflow, DoS bugs at compile time	

Epic 10 – Testnet → Mainnet Launch Sequence

Goal: Safe, audited, incentivized rollout

Story ID	User Story	Acceptance Criteria
LCH-01	Internal devnet (100 nodes) – all epics above	
LCH-02	Public testnet with 10k nodes & real token rewards	
LCH-03	Four independent security audits (Trail of Bits, Kudelski, NCC, academic partner)	
LCH-04	Bug bounty up to $5M	
LCH-05	Genesis with 5-year token emission schedule & useful-work rewards	
LCH-06	Mainnet launch – Day 1 target 100k TPS	

Technical tasks for the user stories under each of the following Epics:

Epic 1 − Enclave-Bound Anonymous Ingress & Mixer Network.

Story ID	Task ID	Detailed Technical Task	Owner (example)	Acceptance / Deliverable	Est. Effort

ING-01	ING-01.01	Design onion transaction format v1 (5-layer, Dilithium-signed outer, ML-KEM ephemeral keys per hop)	Crypto Lead	Protobuf schema + Rust struct + serialization tests	3d
ING-01	ING-01.02	Implement wallet-side onion builder (random relay selection from DHT, layered encryption)	Wallet Team	Unit tests: 100% path coverage, malformed onion rejection	4d
ING-01	ING-01.03	Add transaction size hard cap 3.8 KB serialized (post-onion)	Protocol Lead	Consensus rule + test vectors	1d
ING-02	ING-02.01	Create SGX/SEV/TrustZ one relay enclave binary skeleton (Rust + Graphene/Asylo or Fortanix)	TEE Team	Enclave signs its own measurement, exposes only `process_onion_layer()`	8d

ING-02	ING-02.02	Implement `process_onion_layer(in: EncryptedBlob) → (next_hop_ip:port, forward_blob, attestation_report)` inside enclave	TEE Engineer	No heap allocation > 64 KB inside enclave (constant-time)	6d
ING-02	ING-02.03	Add remote attestation verification library (IAS/AMD VCEK/DCAA) with caching & revocation list	TEE Engineer	Rejects expired or debug-mode enclaves	4d
ING-02	ING-02.04	Implement zero-logging guarantee: enclave wipes all memory on exit, host never persists	Security Lead	Full memory disclosure test passes (Valgrind + custom scanner)	3d

ING-02	ING-02.05	Add chaff/cover traffic generator inside enclave (configurable ratio 1–10×)	TEE Engineer	Uses hardware RNG, timing jitter ±200 ms	3d
ING-02	ING-02.06	Write side-channel mitigation checklist & constant-time crypto (Dilithium & ML-KEM)	Crypto Lead	Passes Spectre/Meltdown & power-analysis test suite	5d
ING-03	ING-03.01	Final relay decrypts inner payload → Halo2 ZK transaction	TEE Engineer	Verifies outer Dilithium sig first	2d
ING-03	ING-03.02	Final relay adds own attestation + monotonic secure timestamp	TEE Engineer	Timestamp sourced from enclave RDTSC + remote sync	2d
ING-03	ING-03.03	Final relay pushes to local shard’s mempool via encrypted Unix socket	Node Team	No plaintext ever touches host memory	2d

ING-03	ING-03.04	Reject double-spend attempts inside enclave before forwarding to mempool	Node Team	Duplicate tx hash → drop + log attestation for slashing	2d
ING-04	ING-04.01	Implement Kademlia DHT extension for attested relays (key = measurement hash)	P2P Team	`find_node` returns only nodes with valid, unexpired attestation	5d
ING-04	ING-04.02	Add DHT bootstrap nodes (10 geographically diverse)	Infra Team	Hard-coded in genesis	1d
ING-04	ING-04.03	Relay registry smart-contract (on-chain) that stakes 10k tokens to advertise	Protocol Lead	Slashable if attestation lies	3d

ING-04	ING-04.04	Wallet periodically refreshes relay list (every 10 min) and prefers low-latency ones	Wallet Team	Ping-based scoring	2d
ING-05	ING-05.01	Implement AI cover-traffic scheduler (LSTM) inside enclave that learns real traffic shape	ML Engineer	Model < 2 MB, runs in < 5 ms	10d
ING-05	ING-05.02	Add configurable padding (Tor Pluggable Transport style) to make all packets 1500 bytes	TEE Engineer	Random dummy bytes from hardware RNG	2d
ING-05	ING-05.03	Implement inter-packet delay jitter engine (±200 ms, normal distribution)	TEE Engineer	Configurable per deployment	1d

ING-06	ING-06.01	Build unlinkability auditor tool (Python + libp2p simulator) that replays full 5-hop paths	Security Researcher	Must achieve k-anonymity ≥ 1,000,000	7d
ING-06	ING-06.02	Run 72-hour traffic capture on testnet and prove no correlation (Pearson < 0.01)	Red Team	Report + graphs	5d
ING-06	ING-06.03	Publish formal anonymity proof (using ProVerif or Tamarin) for 5-hop case	Academic Partner	Peer-reviewed paper (target USENIX Security)	30d (parallel)

Cross-Cutting Tasks (apply to entire Epic)

Task

ID	Task	Deliverable
X-01	CI pipeline for enclave builds (SGX + SEV + TrustZone) with reproducible measurement hashes	GitHub Actions + attestation artefacts
X-02	Fuzzing suite (libFuzzer + AFL++) for onion parser and enclave entry points	99%+ coverage, no crashes after 48h

X-03	Threat model document (STRIDE) + attack tree for mixer	Notion/Miro page
X-04	External TEE security audit (Trail of Bits or similar) focused only on Epic 1	Audit report before testnet

Epic 2 – Verifiable Delay Witnesses & Blind Validation.

Story ID	Task ID	Detailed Technical Task	Owner (example)	Acceptance / Deliverable	Est. Effort
BV-01	BV-01.01	Design Verifiable Delay Witness (VDW) v1 format (Protobuf + fixed layout)	Protocol Lead	Schema: tx_hash(32) + shard_id(8) + lattice_height(8) + embedding_delta_root(32) + tee_sig(96) + metadata ≤ 1800 bytes	2d
BV-01	BV-01.02	Implement VDW generation inside every shard node's TEE after embedding commitment	TEE Engineer	Runs only after 67% neural voting finalises the embedding	4d
BV-01	BV-01	Add inclusion proof from tx → embedding_delt	ZK Engineer	Proof size ≤ 750 bytes, verifies in < 45 ms on desktop	12d

	.03	a using Halo2 recursive circuit			
BV-01	BV-01.04	Bundle VDW = {tx_hash, shard_id, height, inclusion_proof, embedding_root, TEE attestation + signature}	TEE Engineer	Single binary blob, versioned, forward-compatible	3d
BV-01	BV-01.05	Persist every VDW for 5 years in shard-local encrypted DB (AES-GCM-SIV, key derived from enclave)	Node Team	Automatic pruning after 5 years + Merkle history for proofs	4d
BV-02	BV-02.01	Port Halo2 verifier to iOS (Swift + Rust FFI) and Android (Kotlin + JNI)	Mobile ZK Team	Verification time < 80 ms on iPhone 15 / Pixel 9 (measured with Xcode & Android Profiler)	10d
BV-02	BV-02.02	Implement WebAssembly + wasm-gc build of the same verifier	Frontend ZK Team	< 70 ms in Chrome 130 on mid-tier laptop	6d

BV-02	BV-02.03	Build standalone “vdw-verify” CLI tool (Rust) that loads witness from file/QR/base64	Wallet Team	vdw-verify witness.bin → “VALID” or detailed error	3d
BV-02	BV-02.04	Benchmark suite: iOS, Android, Web, Desktop (Intel/Apple Silicon/AMD) – all < 80 ms	Performance Lead	Grafana dashboard + CI enforcement	4d
BV-02	BV-02.05	Add optional TEE-accelerated verification path for devices that have Secure Enclave / Titan M	Mobile TEE Team	Falls back gracefully to pure ZK when unavailable	7d
BV-03	BV-03.01	Implement HTTP/3 + QUIC endpoint /vdw/:tx_hash with range request support	Node Team	Supports byte-range for CDN caching	5d
BV-03	BV-03	Add Cloudflare/Arweave/IPFS gateway integration –	Infra Team	Pinning happens automatically	4d

	.02	every VDW is permanently pinned		within 30 s of generation	
BV-03	BV-03.03	Implement Merkle-ised historical VDW buckets (per day) so old witnesses remain provable after pruning	Node Team	Light clients can still verify 4-year-old txs with < 50 KB extra data	6d
BV-03	BV-03.04	Add rate-limiting + proof-of-work challenge for public VDW endpoints (prevents DoS)	Security Engineer	Adjustable difficulty, cached for 60 s	3d
BV-04	BV-04.01	Extend wallet (mobile + desktop) to auto-fetch and cache VDW when tx is sent	Wallet Team	Stored in encrypted local DB, never deleted unless user explicitly clears	4d
BV-04	BV-04.02	Add “Export Proof” feature → QR code + .vdw file + base64 string	Wallet Team	Works offline after first fetch	3d
BV-04	BV-04	Implement “Show Proof” screen that	Wallet Team	Green checkmark +	3d

	.03	verifies cached VDW on-device without network		“Valid forever” badge	
BV-04	BV-04.04	Add VDW sharing via AirDrop, Nearby Share, and deep-link (hln://proof/…)	Mobile Team	Recipient opens link → instantly validates	5d
BV-05	BV-05.01	Design reorg handling: old VDWs become invalid, new ones issued automatically	Consensus Team	Invalid VDWs return specific error code REORG_INVALID	3d
BV-05	BV-05.02	Implement VDW invalidation Merkle tree per shard (so clients can prove a VDW is revoked)	ZK Engineer	Revocation proof ≤ 1 KB	8d
BV-05	BV-05.03	Add background service that pushes replacement VDWs to known wallets (via push notification	Node + Wallet Team	Users see “Your proof was updated due to reorg”	6d

		+ encrypted channel)			
BV-05	BV-05.04	Write formal specification of VDW finality rules (no reorg can invalidate after 10 cryptographic confirmations)	Protocol Lead	Published as EIP-style document	3d

Cross-Cutting Tasks for Epic 2

Task ID	Task	Deliverable	Est. Effort
X2-01	End-to-end integration test: send tx → receive VDW → verify offline	Automated test in CI that passes on all platforms	5d
X2-02	Fuzzing + property-based testing of VDW parser and verifier	No crashes or false positives after 100 M generated witnesses	6d
X2-03	Formal verification of inclusion proof circuit (using Circom → R1CS → SAPIC model)	ProVerif/Tamarin report proving soundness and hiding	20d (parallel)

X2-04	External audit of VDW format and verification code (focus on Halo2 recursion safety)	Audit report from Kudelski or Trail of Bits	25d (parallel)
X2-05	Benchmark and optimisation target enforcement in CI (< 80 ms)	CI fails if any platform exceeds threshold	3d

Epic 3 – Neural State Embeddings & Homomorphic Updates — the most innovative and mathematically intensive epic of the entire system.

This is the part that replaces Merkle trees and raw state with **512-byte AI-generated latent vectors** that are:

- cryptographically verifiable,
- homomorphically updatable,
- recursively provable,
- losslessly reconstructible via ZK.

Story ID	Task ID	Detailed Technical Task	Owner (example)	Acceptance / Deliverable	Est. Effort
EMB-01	EMB-01.01	Finalise LatentLedger Halo2 circuit specification: maps arbitrary key-value state → fixed 512-byte embedding	ZK + ML Lead	Formal spec (PDF + LaTeX) with exact field elements, constraints count ≤ 8.2 M	5d
EMB-01	EMB-01.02	Implement encoder transformer (24-layer,	ZK-ML Engineer	Runs fully inside Halo2 (no precomputed	18d

		512-dim, GELU, rotary embeddings) in Rust + Halo2 custom gates		lookups), constraint count validated	
EMB-01	EMB-01.03	Implement decoder transformer (symmetric) that reconstructs exact state from embedding + Merkle path	ZK-ML Engineer	Round-trip test: 10^8 random state updates → reconstruction error = 0	16d
EMB-01	EMB-01.04	Generate trusted setup (Powers of Tau + phase 2) for LatentLedger circuit with 2^{28} constraints	Ceremony Team	MPC ceremony with 100+ participants, toxic waste destroyed, verified on IPFS	14d (parallel)
EMB-01	EMB-01.05	Write soundness proof (knowledge-soundness + simulation-extractability) for the full encoder/decoder pair	Academic Cryptographer	Published paper (target Crypto/IACR)	30d (parallel)

EMB-02	EMB-02.01	Design homomorphic delta format: 512-byte vector that can be added to any embedding to get embed(new_state)	ZK Engineer	$\delta = \text{embed}(S \cup \{k \to v+\Delta\}) - \text{embed}(S)$ works for balance transfers up to 2^{128}	6d
EMB-02	EMB-02.02	Implement HomomorphicUpdate Halo2 circuit: embedding_prev + delta → embedding_new + proof	ZK Engineer	Proof size ≤ 380 bytes, verifies in < 25 ms	12d
EMB-02	EMB-02.03	Prove homomorphism is complete for supported operations (transfer, mint, burn)	ZK Engineer	Formal proof + 10^7 random test vectors	8d
EMB-02	EMB-02.04	Add batched homomorphic updates (up to 256 txs in one delta)	ZK Engineer	Reduces proof size per tx to ~1.5 bytes	10d

EMB-03	EMB-03.01	Generate synthetic training dataset: 500 M realistic tx sequences (payments, DeFi, NFTs)	Data Engineer	Stored in Parquet on S3, 2 TB total	7d
EMB-03	EMB-03.02	Train initial compressor model using PyTorch + DeepSpeed on 64 × H100 (target < 1e-9 reconstruction loss)	ML Team	Model checkpoint < 8.2 MB (quantised int4 + Huffman)	21d
EMB-03	EMB-03.03	Convert trained model → Halo2 custom gates (full arithmetic circuit transcription)	ZK-ML Engineer	No accuracy loss vs floating-point model	25d
EMB-03	EMB-03.04	Implement model distillation loop: on-chain model is retrained every 30 days using federated gradients (Epic 6)	ML Engineer	Model size stays ≤ 8 MB forever	10d

EMB-03	EMB-03.05	Benchmark compression: 100 kB state → 512 bytes (≥ 195× compression)	Performance Lead	Grafana dashboard + CI enforcement	4d
EMB-04	EMB-04.01	Design recursive inclusion proof: tx → homomorphic delta → final embedding root	ZK Engineer	Uses Halo2 recursion + Plonkish arithmetization	10d
EMB-04	EMB-04.02	Implement EmbeddingInclusion circuit (proof size target ≤ 800 bytes)	ZK Engineer	Verifies in < 60 ms on iPhone 15	18d
EMB-04	EMB-04.03	Add aggregation: 256 inclusion proofs → one aggregated proof	ZK Engineer	For light clients receiving batched updates	12d
EMB-04	EMB-04.04	Write Nova-style folding scheme fallback for mobile devices	ZK Researcher	Optional path, < 90 ms verification	15d

		(if recursion too slow)			
EMB-05	EMB-05.01	Implement periodic full-shard re-embedding inside TEE (every 1000 txs or 10 s)	TEE + Node Team	Runs encoder on current state trie, commits new embedding root	6d
EMB-05	EMB-05.02	Implement embedding root commitment in consensus layer (67% of nodes must agree on new root)	Consensus Team	Part of neural voting (Epic 4)	4d
EMB-05	EMB-05.03	Add emergency “slow path” fallback: if model diverges > 1e-6, fall back to traditional Merkle root for one epoch	Safety Engineer	Automatic detection + governance alert	5d
EMB-05	EMB-05.04	Implement on-chain model upgrade ceremony with 30-day delay + staking vote	Protocol Lead	New encoder/decoder pair can be swapped without	8d

		breaking history	

Cross-Cutting Tasks for Epic 3 (the hardest epic)

Task ID	Task	Deliverable	Est. Effort
X3-01	End-to-end golden test: 1 M txs → final embedding → full state reconstruction → exact match	CI test that must never break	10d
X3-02	Continuous constraint optimisation: target ≤ 8.2 M constraints for encoder+decoder combined	Weekly benchmark + Halo2 gate specialisation	ongoing
X3-03	External audit of LatentLedger circuit (focus: soundness, no backdoors, correct homomorphism)	Trail of Bits + academic partner audit report (minimum 12 weeks)	90d (parallel)
X3-04	Formal verification of homomorphic property using Coq or Lean	Machine-checked proof	120d (parallel)
X3-05	Red-team “embedding poisoning” exercises (try to make two different states collide)	Report + mitigations	14d

X3-06	Model quantisation + compression pipeline (int4 + Huffman) to fit in TEEs and mobile			Final model size ≤ 7.8 MB	10d

Epic 4 – AI-Native Optimistic Consensus

Story ID	Task ID	Detailed Technical Task	Owner (example)	Acceptance / Deliverable	Est. Effort
CON-01	CON-01.01	Define neural vote message format (Protobuf): node_id, predicted_embedding_hash[32], partial_BLS_sig[48], reputation_score, TEE attestation	Consensus Lead	Versioned, ≤ 256 bytes total	2d
CON-01	CON-01.02	Implement on-device distilled inference model (≤ 2 MB) that predicts next embedding from current embedding + batched homomorphic deltas	ML + ZK Engineer	Runs in < 8 ms on Ryzen 7950X and Apple M2, outputs exactly 512-byte embedding hash	14d

CON-01	CON-01.03	Integrate prediction model into node TEE — model weights loaded once at startup, never leave enclave	TEE Engineer	Remote attestation proves correct model hash	6d
CON-01	CON-01.04	Nodes broadcast signed neural vote within 150 ms of receiving a valid batch	Node Team	Measured 99-th percentile < 180 ms on 5-continent testnet	5d
CON-02	CON-02.01	Implement weighted threshold aggregation: sum(stake × reputation × vote) ≥ 67 % of total active weight → instant probabilistic finality	Consensus Engineer	Uses BLS aggregate signatures (BLS12-381)	8d
CON-02	CON-02.02	After 67 % agreement, commit embedding root + full BLS threshold signature to lattice history	Consensus Engineer	Finality event emitted, VDWs become issuable	4d

CON-02	CON-02.03	Implement “fast-path” gossip acceleration: only embedding hash + partial sigs are gossiped (full txs follow lazily)	P2P Team	Reduces bandwidth 40× during normal operation	6d
CON-02	CON-02.04	Add configurable fast-finality threshold per shard (default 67 %, governance can raise to 80 % in high-attack periods)	Protocol Lead	On-chain parameter, 14-day delay	3d
CON-03	CON-03.01	Design challenge phase protocol: any node can open a challenge within 800 ms of a fast-final block	Consensus Engineer	Challenge bond = 0.5 % of staked amount	4d
CON-03	CON-03.02	Implement Monte-Carlo dispute resolution inside TEE	TEE + ML Engineer	Resolves in < 650 ms, outputs winning embedding root	18d

		cluster: 10 000 simulated executions using sampled randomness			
CON-03	CON-03.03	Implement fraud proof generation: losing side must provide Halo2 proof of misbehaviour within 2 s or get slashed	ZK Engineer	Slash 1–5 % of stake (linear to confidence discrepancy)	12d
CON-03	CON-03.04	Add economic finality timer: after 1.8 s with no successful challenge → cryptographic finality (irreversible even with 60 % attack)	Consensus Lead	Proven in game-theoretic security paper (published internally)	5d
CON-03	CON-03.05	Simulate 10 000 attack scenarios (33 % malicious, 20 % latency, eclipse, etc.) and prove liveness & safety	Red Team + Researcher	Report + fixes before testnet	21d

CON-04	CON-04.01	Implement reputation oracle inside TEE: continuously ingests federated gradients + vote honesty → updates 256-bit reputation score	ML + TEE Engineer	Score ∈ [0, 1], persisted encrypted, updated every 10 min	10d
CON-04	CON-04.02	Effective voting weight = stake × reputation (multiplicative)	Consensus Engineer	Reputation < 0.1 → vote weight = 0 (auto-exit)	3d
CON-04	CON-04.03	Add reputation recovery mechanism: honest nodes recover 0.02/week after punishment	Protocol Lead	Prevents permanent exile for temporary faults	2d
CON-04	CON-04.04	Reputation slashing for provable equivocation (double-voting)	Security Engineer	Immediate 50 % reputation burn + 7-day jail	4d
CON-05	CON-05.01	Deploy 500-node global testnet (5 continents,	Testnet Ops	Runs 30 days continuously	14d

		real internet latency) with automated chaos (packet loss, partitions, byzantine faults)			
CON-05	CON-05.02	Instrument and enforce finality KPIs in CI: median 600 ms probabilistic, 1.8 s crypto, > 99.999 % correct under 33 % byzantine	Performance + QA	Grafana + alerting dashboard, CI gate	8d
CON-05	CON-05.03	Run long-running stress test: 10 M txs at 200 k TPS sustained, measure finality distribution	QA + Infra	Final report + tuning parameters	10d
CON-05	CON-05.04	Publish security & performance paper (target IEEE S&P or USENIX Security)	Researcher	Peer-reviewed before mainnet	60d (parallel)

Cross-Cutting Tasks for Epic 4

Task ID	Task	Deliverable	Est. Effort
X4-01	End-to-end golden path test: tx → neural vote → 67 % → instant finality → VDW issued	Must pass 100 % in CI forever	7d
X4-02	Formal BFT proof under partial synchrony + AI predictor model	Lean/Coq formalisation (or at least detailed game-theoretic proof)	90d (parallel)
X4-03	External consensus audit (least three firms: Trail of Bits, Runtime Verification, Informal)	Clean audit reports before testnet launch	12 weeks (parallel)
X4-04	Fuzzing + fault injection suite for neural vote messages and BLS aggregation	No crashes or invalid finality after 100 M malformed messages	10d
X4-05	Chaos-monkey automation (Netem + tc + custom byzantine actors)	Runs 24/7 on testnet	8d

Epic 5 – Dynamic Neural Sharding (needs working embeddings + consensus).

Story ID	Task ID	Detailed Technical Task	Owner (example)	Acceptance / Deliverable	Est. Effort

SHD-01	SHD-01.01	Finalise shard load metrics definition: TPS, gas/s, embedding update size, cross-shard tx rate, 95th-pct latency	Consensus + ML Lead	Published spec v1, used everywhere	2d
SHD-01	SHD-01.02	Implement per-shard time-series collector (Prometheus → VictoriaMetrics) with 1-second granularity	Observability Team	Metrics: shard_tps, shard_gas_per_sec, cross_shard_ratio, p95_finality_ms	5d
SHD-01	SHD-01.03	Train LSTM load predictor (input: last 120 s of 12 metrics, output: probability of overload in next 15 s)	ML Engineer	Model < 1.2 MB, inference < 4 ms on CPU, > 95 % accuracy on 30-day testnet data	18d
SHD-01	SHD-01.04	Deploy predictor inside every node's TEE (updated weekly via Epic 6 federated learning)	TEE + ML Engineer	Remote attestation proves correct model version	6d

SHD-01	SHD-01.05	Add predictive alert → any node can propose split/merge 15 s before overload	Node Team	Alert bonded with 0.1 % stake (slashed if false positive > 5 %)	4d
SHD-02	SHD-02.01	Design state embedding bisection algorithm: deterministically split one 512-byte embedding into two valid child embeddings	ZK + ML Engineer	Uses fixed random seed derived from shard_id + height, proven lossless in round-trip tests	10d
SHD-02	SHD-02.02	Implement SplitProposal message: old_shard_id → new_shard_A + new_shard_B, both with new embedding roots	Consensus Engineer	Requires 67 % of current shard stake to sign	5d
SHD-02	SHD-02.03	Execute live split inside TEEs: re-execute last N txs on both child shards to reach identical	TEE Engineer	Split finalises in < 4 seconds (measured on 10 k TPS shard)	12d

		embedding roots			
SHD-02	SHD-02.04	Migrate in-flight txs and mempool entries to correct child shard automatically	Node Team	No tx lost or reordered	6d
SHD-02	SHD-02.05	Update DHT + relay routing tables instantly on split (new shard_ids advertised globally)	P2P Team	Light clients see new shards within 2 gossip rounds	5d
SHD-02	SHD-02.06	Add rollback protection: if split fails quorum, revert to old embedding root in next block	Consensus Engineer	Tested with 30 % byzantine nodes failing to acknowledge split	7d
SHD-03	SHD-03.01	Implement merge trigger: two shards both < 10 TPS sustained for 10 minutes → automatic merge proposal	Node Team	Proposal bonded, requires 67 % stake from both shards	4d

SHD-03	SHD-03.02	Design merge embedding algorithm: combine two 512-byte embeddings → one parent embedding (reverse of bisection)	ZK + ML Engineer	Lossless, deterministic, same seed method	8d
SHD-03	SHD-03.03	Execute live merge inside TEEs with coordinated checkpoint at same lattice height	TEE Engineer	Merge completes in < 6 seconds	10d
SHD-03	SHD-03.04	Retire old shard_ids and purge state after 1000 blocks of inactivity	Node Team	Frees disk space automatically	3d
SHD-04	SHD-04.01	Implement Reed–Solomon erasure coding (k=5, m=2 → 7 total replicas) for every shard state embedding and recent 10 000 txs	Distributed Systems Lead	Survives 40 % node loss with zero downtime	14d

SHD-04	SHD-04.02	Build AI placement optimiser that minimises cross-region replication + latency (genetic algorithm, runs every 10 min)	ML + Infra Engineer	Reduces average cross-shard finality from 800 ms → 320 ms	16d
SHD-04	SHD-04.03	Implement repair protocol: missing chunk detected → reconstruct from 5 surviving pieces → push to new node	Node Team	Repair time < 8 s per shard	7d
SHD-04	SHD-04.04	Add geo-aware placement constraints (max 2 replicas in same AWS region)	Infra Team	Enforced via on-chain validator metadata	4d
SHD-04	SHD-04.05	Test 500 simultaneous shard splits + 40 % node crash → full recovery in < 30 s	Chaos Team	Must pass before mainnet	10d

Cross-Cutting Tasks for Epic 5

Task ID	Task	Deliverable	Est. Effort
X5-01	End-to-end integration test: 100 → 800 → 200 shards in 30 minutes under real traffic	Video + metrics proof	10d
X5-02	Formal proof of liveness & safety under dynamic membership (extends existing BFT proofs)	Internal paper + external academic review	60d (parallel)
X5-03	Fuzzing + property tests for split/merge embedding math	No collisions or invalid embeddings after 10^8 trials	12d
X5-04	External audit of sharding logic (Runtime Verification + one more firm)	Clean report before testnet	10 weeks (parallel)
X5-05	Real-time visualisation dashboard (shards as 3D lattice, live splits/merges, replication map)	Public testnet explorer feature	14d
X5-06	Chaos engineering suite: random splits, merges, mass node kills, network partitions	Runs 24/7 on staging testnet	8d

Epic 6 – Useful-Work Economy & Federated Learning (the economic engine that keeps the AI improving forever).

Story ID	Task ID	Detailed Technical Task	Owner (example)	Acceptance / Deliverable	Est. Effort
UW-01	UW-01.01	Design encrypted gradient format (256 KB max, int8 quantised, COSE_Encrypt0 with node's Dilithium key)	ML-Crypto Engineer	Protobuf schema + Rust + PyTorch serialization	4d
UW-01	UW-01.02	Implement secure aggregation protocol inside TEE cluster (additive homomorphic masking with verifiable secret sharing)	TEE + Crypto Engineer	10 000 nodes → final aggregated gradient in < 8 s, zero individual gradient leakage	21d
UW-01	UW-01.03	Node-side gradient generation hook: after every 1000 txs (or 15 s), run local training step on anonymised tx batch → encrypted gradient	Node + ML Engineer	Gradient size ≤ 240 KB, runs in < 900 ms on 16-core validator	12d

UW-01	UW-01.04	Add privacy filter: DP-SGD with noise σ=0.5 + per-example clipping at 1e-6 before encryption	Privacy Engineer	Proven (ε,δ)-DP bounds published, passes Google's Opacus verification	8d
UW-01	UW-01.05	Implement drop-out tolerance: aggregation succeeds with ≥ 70 % of expected gradients	TEE Engineer	No blocking, late gradients accepted in next round	4d
UW-02	UW-02.01	Build parameter server TEE cluster (16–32 global instances, Intel SGX + AMD SEV + ARM CCA)	TEE Ops	Runs secure aggregation + model averaging, remote attestation required for every connection	18d
UW-02	UW-02.02	Implement model update pipeline: every 10 minutes → decrypt aggregate → average → test on hold-out set → publish new	ML Engineer	New model versioned on-chain (IPFS + embedding hash)	10d

		version if Δloss > 0.003			
UW-02	UW-02.03	Add model validation oracle: 100 independent staked validators re-run inference on fixed test set → must match server within 1e-6	Consensus Team	Prevents poisoned updates	7d
UW-02	UW-02.04	Automatic rollback to previous model if validation fails or network finality drops > 20 %	Safety Engineer	Triggered within 2 minutes	5d
UW-03	UW-03.01	Design on-chain model registry contract (move-style): stores IPFS CID + embedding root + Dilithium-signed metadata	Protocol + Move Engineer	Immutable, versioned, 7-day governance delay for upgrades	8d

UW-03	UW-03.02	Implement staking-weighted voting for major model upgrades (new architecture, not just weights)	Governance Team	Requires 67 % of total stake + 30-day voting period	6d
UW-03	UW-03.03	Add model upgrade ceremony: new model must be accompanied by Halo2 proof that it preserves homomorphic properties	ZK + ML Engineer	Proof size ≤ 2 MB, verifies in < 3 s	14d
UW-03	UW-03.04	Publish every model version permanently on Arweave + Filecoin	Infra Team	10-year guaranteed availability	4d
UW-04	UW-04.01	Implement Shapley-value approximation for gradient contribution (Last-Value + periodic full TMA)	Incentives Economist	Accuracy > 98 % vs exact Shapley on 10 k-node samples	16d

UW-04	UW-04.02	Deploy on-chain micro-payments: every accepted gradient pays 0.02–0.15 HLN tokens (proportional to Shapley contribution)	Tokenomics + Node Team	Payments batched every 10 min, paid from inflation pool	10d
UW-04	UW-04.03	Add anti-gaming measures: gradient similarity clustering → slash clones > 95 % identical	Security Engineer	Tested with 30 % Sybil attack → > 99 % detection	9d
UW-04	UW-04.04	Implement data-oracle work rewards: running price feed inference, fraud detection models, etc., counts as 3× normal gradient work	Oracle Team	Pays extra from oracle fee pool	7d
UW-04	UW-04.05	Economic simulation: 100 k nodes, 5 years → prove inflation < 4 %/year and	Economist + Simulator	Published model + open-source simulator	21d

useful-work
dominates
energy spend

Cross-Cutting Tasks for Epic 6

Task ID	Task	Deliverable	Est. Effort
X6-01	End-to-end test: 50 000 nodes submitting gradients → new model every 10 min → measurable improvement weekly	Live on public testnet for 90 days	30d
X6-02	Formal privacy proof of the full federated pipeline (DP + secure aggregation)	(ε=1.2, δ=1e-8) per 30 days, audited by differential privacy experts	45d (parallel)
X6-03	External audits: secure aggregation (CrypTFlow2 team or Galois), incentives (leastauthority)	Two clean reports before mainnet	12 weeks (parallel)
X6-04	Public bug bounty for gradient poisoning / model stealing	Up to $1 M rewards	ongoing
X6-05	Real-time AI dashboard: model accuracy, contribution leaderboards, privacy budget remaining	Public explorer page	14d

X6-06	Token emission schedule + useful-work treasury contract		Audited Move/Rust code, 10-year curve	10d

Epic 7 – Quantum-Resistant & Future-Proof Cryptography Suite.

This is a “pure crypto” epic — every primitive is NIST-approved, round-3 or later, and ready for quantum attacks today.

Story ID	Task ID	Detailed Technical Task	Owner (example)	Acceptance / Deliverable	Est. Effort
QR-01	QR-01.01	Replace all ECDSA / EdDSA signatures with **CRYSTALS-Dilithium 3** (security level 3, ~3.3 KB sig, 1.8 KB pk)	Crypto Lead	Full drop-in replacement interface SignatureScheme	5d
QR-01	QR-01.02	Integrate official NIST submission reference code (C + assembly) + Rust bindings via	Crypto Engineer	100 % test vectors pass (NIST KATs + Wycheproof)	7d

		dilithium-crystals crate			
QR-01	QR-01.03	Optimise Dilithium-3 sign/verify for x86-64 AVX2 and ARM Neon — target < 60 µs verify on Intel Ice Lake, < 90 µs on Apple M2	Optimisation Engineer	Benchmarks published, CI enforces thresholds	14d
QR-01	QR-01.04	Add constant-time hardened implementation (no secret-dependent branches, no table lookups)	Security Engineer	Passes dudect + FlowTracker + ctgrind	8d
QR-01	QR-01.05	Replace every on-chain and P2P signature (block headers, votes, attestations, transactions, VDWs) with Dilithium-3	Node + Protocol Team	No remaining secp256k1 / ed25519 anywhere in critical paths	6d
QR-02	QR-02.01	Implement **ML-KEM-768** (formerly Kyber-768) for	Crypto Engineer	Replaces X25519 everywhere (onion routing,	6d

		all enclave-to-enclave and node-to-node key exchange		QUIC, TEE channels)	
QR-02	QR-02.02	Hybrid post-quantum handshake: ML-KEM-768 + X25519 (for forward secrecy until 2035)	Crypto Lead	Dual-KEM construction per RFC draft-ietf-tls-hybrid-design	5d
QR-02	QR-02.03	Port liboqs (Open Quantum Safe) v0.12+ into node, enclave, and mobile builds	Integration Engineer	Single compile-time flag pq_crypto = true activates everything	7d
QR-02	QR-02.04	Benchmark handshake latency impact — target < +12 ms vs classical X25519	Performance Team	Real 4G/5G + satellite tests	4d
QR-03	QR-03.01	Add **SPHINCS+-SHA256-192s-robust** as stateless, hedge backup signature scheme for cold	Crypto Engineer	Signature size ~41 KB, verify < 1.2 s on desktop — used only when maximum caution needed	6d

		wallets and genesis keys			
QR-03	QR-03.02	Implement “hedged signing” mode: normal Dilithium + optional SPHINCS+ co-signature for high-value txs	Wallet Team	User toggle “Quantum doomsday mode”	4d
QR-03	QR-03.03	Pre-generate and store 10 000 SPHINCS+ keypairs in HSMs for foundation recovery keys	Ops + Security	Keys never leave HSM, public keys published at genesis	3d
QR-04	QR-04.01	Design cryptographic agility framework: every signature and KEM tagged with CryptoVersion enum	Protocol Lead	New enum values can be added without hard fork	5d
QR-04	QR-04.02	Implement on-chain “Crypto Upgrade” governance proposal type	Governance Team	Example: upgrade Dilithium-3 → Dilithium-5 in 2032 without	8d

		— 180-day voting + 90-day migration period		breaking old signatures	
QR-04	QR-04.03	Add backwards compatibility layer: nodes continue to verify old ECDSA/EdDSA signatures for 24 months after launch	Node Team	“Legacy mode” disabled via governance in year 3	6d
QR-04	QR-04.04	Build migration tooling for wallets: one-click “Upgrade all keys to PQ” with batched transaction	Wallet Team	Zero-downtime, works offline after first sync	10d
QR-04	QR-04.05	Publish cryptographic continuity plan 2025–2040 with concrete upgrade triggers (e.g., NIST announces new standard, IBM 10 000-qubit, etc.)	Crypto + Governance	Signed PDF, on-chain immutable copy	7d

Cross-Cutting Tasks for Epic 7 (mostly parallelisable)

Task ID	Task	Deliverable	Est. Effort
X7-01	Full NIST + Wycheproof + Project Wycheproof test vectors for every primitive	CI must pass 100 % forever	7d
X7-02	External cryptography audit of all new primitives and integrations (PQShield + Kudelski or QuSecure)	Two independent clean reports	10 weeks (parallel)
X7-03	Formal verification of Dilithium-3 and ML-KEM-768 constant-time implementations using Jasmin or Fiat-Crypto	Machine-checked proofs	90d (parallel)
X7-04	HSM + secure enclave integration for Dilithium private keys (AWS CloudHSM, Azure Dedicated HSM, YubiHSM)	Production-grade key protection for validators	14d
X7-05	Quantum threat monitoring dashboard (tracks Shor-capable qubit counts, lattice attack papers, etc.)	Public page + governance alerts	10d
X7-06	Emergency “Quantum Break” hard fork playbook (activate SPHINCS+ everywhere in < 72 h)	Tested on staging, signed by foundation	5d

Epic 8 – Extreme Scalability Layer

Story ID	Task ID	Detailed Technical Task	Owner (example)	Acceptance / Deliverable	Est. Effort
SCL-01	SCL-01.01	Implement fully parallel shard execution engine: 500+ shards run simultaneously on one node using Tokio async tasks + Rayon thread pool	Runtime Team	Single 64-core validator sustains 180 k TPS locally (measured with 500 shards)	10d
SCL-01	SCL-01.02	Add per-shard memory isolation (Linux cgroups + Rust jemalloc arenas)	Systems Engineer	One misbehaving shard cannot OOM the node	6d
SCL-01	SCL-01.03	Cross-shard messaging via asynchronous “mailbox” (zero-copy, lock-free ring buffers)	Runtime Team	Cross-shard tx finality ≤ 1.1 s (99-th percentile) on 5-continent testnet	12d
SCL-01	SCL-01.04	Automatic load-balancing of shards across	Performance Lead	CPU utilisation stays 85–95 % on 128-core machines	8d

		CPU/NUMA nodes using work-stealing scheduler			
SCL-02	SCL-02.01	Implement native account abstraction (EIP-4337 style but baked into genesis)	Protocol + Wallet Team	Users never sign gas payments; any token or sponsor can pay	7d
SCL-02	SCL-02.02	Add paymaster marketplace contract (Move/Rust) — third parties compete to sponsor txs	Smart Contract Team	Top 10 paymasters cover > 95 % of new-user txs on testnet	9d
SCL-02	SCL-02.03	Bundler service (MEV-resistant) : aggregates 5 000 user-ops per second, pays gas in HLN, rebates in any ERC-20	Bundler Team	Average user pays 0 gas for first 90 days	10d
SCL-02	SCL-02.04	One-click “Gasless mode” in all reference wallets (mobile + web)	Wallet Team	Toggle works offline, uses cached paymaster signatures	5d

SCL-03	SCL-03.01	Build AI fee predictor microservice (runs same LSTM from sharding epic) — returns exact fee 10 s into the future	ML Engineer	Accuracy > 99.9 % on 30-day testnet data	8d
SCL-03	SCL-03.02	Integrate fee predictor into wallet UX — shows “Your tx will cost exactly 0.00007 HLN” before signing	Wallet Team	Users see deterministic price, never overpay	4d
SCL-03	SCL-03.03	Eliminate MEV completely: priority = exact fee paid (no tip); sequencer cannot reorder profitably	Consensus + Protocol	Prove with 100 k simulated txs that extractor profit = 0	6d
SCL-03	SCL-03.04	Add “Fee smoothing” treasury that refunds overpays when congestion drops	Tokenomics Team	Users get automatic micro-refunds within 10 min	5d

SCL-04	SCL-04.01	Enable recursive embedding compression for all proofs (VDWs, inclusion proofs, fraud proofs) → target 900× smaller than Polygon zkEVM	ZK Engineer	Average private transfer proof drops from 380 KB → 420 bytes	14d
SCL-04	SCL-04.02	Implement Nova-style folding for mobile light clients (fallback when full recursion too slow)	ZK-Mobile Engineer	Verification time < 60 ms on iPhone 15 with 10× compression	16d
SCL-04	SCL-04.03	Add proof aggregation nodes (specialised validators) that fold 10 000 proofs into one	ZK Infra Team	Reduces light-client sync data from 800 MB → 80 KB per day	12d
SCL-04	SCL-04.04	Benchmark end-to-end: private payment → full validation on fresh device in	Performance + Mobile	Public leaderboard + CI enforcement	7d

	< 1.2 s with < 2 KB data		

Cross-Cutting Tasks for Epic 8

Task ID	Task	Deliverable	Est. Effort
X8-01	1 M+ TPS stress test (7 days continuous, 5 continents, real DeFi + NFT load)	Public video + on-chain transaction explorer proof	21d
X8-02	Global benchmark suite (AWS i4i.32xlarge, Hetzner AX162, MacStudio, Pixel 9) — all > 500 k TPS/node	Published table + open-source benchmark tool	14d
X8-03	External performance + correctness audit of parallel execution engine (Runtime Verification)	Clean report before mainnet	8 weeks (parallel)
X8-04	Gasless onboarding campaign framework (paymasters pre-funded with 100 M HLN)	10 M gasless wallets in first 30 days target	10d
X8-05	Real-time scalability dashboard (live TPS, shard count, proof size, fee graph)	Public explorer page	10d
X8-06	Emergency “Throttle” governance parameter (can cap TPS at 200 k if consensus bugs appear)	Tested and documented	4d

Epic 9 – Developer & User Experience Layer

Story ID	Task ID	Detailed Technical Task	Owner (example)	Acceptance / Deliverable	Est. Effort
DX-01	DX-01.01	Release **hln-sdk** v1 for Rust, TypeScript/JavaScript, Python, Go, Swift, Kotlin — all identical APIs	SDK Team	npm i @hln/sdk, pip install hln-sdk, etc. — all compile and pass same 500 e2e tests	18d
DX-01	DX-01.02	Built-in ZK & embedding generation in SDK — sendPrivateTransfer(to, amount) auto-generates onion + VDW + proof	ZK + SDK Engineer	Zero extra code needed for full privacy	10d
DX-01	DX-01.03	Auto-paymaster selection — SDK picks cheapest live paymaster for user	SDK + Wallet Team	New users pay $0 gas forever without config	5d

DX-01	DX-01.04	Generate OpenAPI 3.1 spec + Postman collection + GraphQL endpoint for all node JSON-RPC methods	DevRel Engineer	Developers can use Insomnia/Thunder Client directly	7d
DX-02	DX-02.01	Implement **hln deploy** CLI (one binary, works on macOS/Linux/Windows) — hln deploy ./contract compiles, proves, deploys private contract	Tooling Team	Single command, < 6 s to mainnet deployment, prints QR code for contract address	12d
DX-02	DX-02.02	Support Move → Halo2 circuit → recursive proof pipeline (private state by default)	Move + ZK Engineer	#[private] struct Balance just works	21d
DX-02	DX-02.03	Add VS Code extension with syntax highlighting, autocomplete, inline proof size estimator,	DevRel + IDE Team	50 k downloads in first month target	14d

		“Deploy to HLN” button			
DX-02	DX-02.04	One-click testnet faucet in CLI and web — gives 100 HLN + gasless paymaster credit	DevRel Team	hln faucet → instant balance	4d
DX-03	DX-03.01	Release **one-click light client** for iOS, Android, Chrome/Edge/Firefox, Safari (all < 100 KB sync)	Mobile + Web Team	First open → full security in < 8 s, works forever offline after that	21d
DX-03	DX-03.02	Web light client as WebAssembly + COOP/COEP headers — works in any iframe	Web Team	Can be embedded in any dApp with one <script> tag	10d
DX-03	DX-03.03	Deep-link + QR scheme hln://pay/addr/amount and hln://proof/xyz — opens wallet instantly	Mobile Team	Scan QR → wallet opens → tx ready to sign in < 800 ms	8d

DX-03	DX-03.04	Progressive Web App wallet with biometric login + recovery via iCloud/Keychain	Mobile + Security	No seed phrase ever shown to user	12d
DX-04	DX-04.01	Implement on-chain AI auditor model (8 MB transformer) that scans every contract at deploy time for 95 %+ of known vulnerability classes	ML + ZK Engineer	Reentrancy, overflow, DoS, timestamp dependence → flagged with explanation	25d
DX-04	DX-04.02	Add “Fix this for me” button — AI suggests patched version, user clicks Approve	DevRel + AI Team	80 %+ of common bugs auto-fixed in < 3 s	14d
DX-04	DX-04.03	Publish public vulnerability leaderboards and bug bounty integration	Security + DevRel	$5 M total bounty pool, top 100 auditors ranked	10d

DX-04	DX-04.04	Formal verification bridge — one-click export to Certora / Act / KEVM for contracts that need 100 % proof	Formal Methods Team	Used by top 10 DeFi teams within 30 days of launch	16d

Bonus User-Facing Stories (included in Epic 9)

Story ID	Task	Deliverable	Est. Effort
DX-05	“Send me $10 privately” natural UX — recipient never sees sender address, just scans QR or clicks link	Works in every reference wallet	7d
DX-06	Private ENS — .hln names that resolve to shielded addresses only visible to owner	vitalik.hln → shielded z-address, no public linkability	12d
DX-07	Bridge UX that preserves privacy — ETH → private HLN, Solana → private HLN without ever de-anonymizing	One-click in wallet, < 15 s, zero address reuse	18d
DX-08	Mobile push notifications for private incoming payments (without revealing amount or sender)	“You received money” → open wallet → reveal only after biometric unlock	8d

Cross-Cutting Tasks for Epic 9

Task ID	Task	Deliverable	Est. Effort
X9-01	100 % e2e test coverage across all SDK languages and platforms	CI breaks if anything fails on any platform	14d
X9-02	Public documentation site (docusaurus) + interactive playground	docs.hln.net — 10 k visits/week in first month	21d
X9-03	Hacker One / Immunefi bug bounty program launch + $10 M total pool	Live at testnet launch	7d
X9-04	Top 50 Ethereum/Solana dApps get one-click “Port to HLN” kit + $50 k grant each	15+ major dApps live in first 90 days	30d
X9-05	Global developer bootcamp tour (Singapore, Seoul, Berlin, NYC, Dubai, Buenos Aires)	5 000+ developers trained in person + recordings	60d (parallel)

Epic 10 – Testnet → Mainnet Launch Sequence (security audits, genesis ceremony, token economics, final chaos testing, and the actual launch).

Story ID	Task ID	Detailed Task (with exact deliverables & owners)	Owner	Success Criteria / Artefact	Calendar (weeks before launch)	Est. Effort
LCH-01	LCH-01.01	Spin up closed **Internal Devnet-Alpha** with 100–200 core team nodes (all epics 1–9 integrated)	Core Tech + Ops	30-day continuous run, ≥ 300 k TPS, zero crashes, all VDWs valid	T-60	4w
LCH-01	LCH-01.02	Run 168-hour **chaos campaign** (random node kills, 50 % byzantine, latency injections, full partition healing)	Chaos + QA	System self-heals within SLA every time; final report published	T-58	3w
LCH-01	LCH-01.03	Freeze feature development → **code complete** for mainnet	CTO	Git tag v1.0.0-rc1, no new features merged	T-56	1d

				after this date		
LCH-02	LCH-02.01	Launch **Public Testnet-Omega** with real economic incentives (10 M HLN faucet + useful-work rewards)	Testnet Ops + Token Team	≥ 10 000 independent nodes in week 1, ≥ 100 k daily active wallets by week 8	T-52	8w total
LCH-02	LCH-02.02	Run three staged **load & chaos weeks**: 500 k TPS → 1 M TPS → 1.5 M TPS with real DeFi/NFT/GameFi dApps	Community + Ops	Public leaderboard, no rollback needed	T-52 to T-44	8w
LCH-02	LCH-02.03	Incentive program: top 100 validators by stake + honesty get 5–20× reward multiplier	Tokenomics	Achieves geographic & hardware diversity (no >8 % in one AWS region)	T-52	4w

LCH-02	LCH-02.04	Run **“Genesis Rehearsal”** — full dry-run of mainnet genesis ceremony on testnet	Foundation + Core Tech	100 % success, recorded & livestreamed	T-46	2w
LCH-03	LCH-03.01	Commission **four parallel security audits** (minimum): • Trail of Bits (full system) • Kudelski Security (crypto + TEE) • Runtime Verification (formal verification of consensus + embeddings) • Academic partner (UC Berkeley or ETH Zurich)	Security Lead	All four final reports **CLEAN or LOW only** — no critical/high unfixed	T-50 → T-28	22w (parallel)

LCH-03	LCH-03.02	Fix every finding → re-audit rounds until zero critical/high	Core Tech	Public audit reports + fix commits	T-28 → T-20	8w
LCH-03	LCH-03.03	Publish all four final audit reports + attestation letters publicly	Foundation	On hln.net/security day of publication	T-20	1w
LCH-04	LCH-04.01	Launch **$10 M+ bug bounty** on Immunefi (Critical: up to $5 M, Quantum-break: $10 M)	Security + Foundation	Live 90 days before genesis, at least 500 whitehats registered	T-40	4w
LCH-04	LCH-04.02	Run **red-team penetration test** (NCC Group or Cossack Labs) with full insider access	External Red Team	Final report clean or cosmetic only	T-32 → T-24	8w

LCH-04	LCH-04.03	Pay out any valid critical bounties before genesis (public transparency)	Foundation	Zero unresolved critical bugs at genesis	Ongoing	—
LCH-05	LCH-05.01	Final tokenomics & emission schedule (10-year curve, useful-work treasury, staking APY model)	Tokenomics + Governance	Signed PDF + on-chain immutable contracts	T-36	6w
LCH-05	LCH-05.02	Genesis allocation ceremony (multi-sig + TEE + HSM) — foundation, early contributors, ecosystem fund, useful-work treasury	Foundation + Legal	4096 SPHINCS+ + Dilithium key shares generated in audited ceremony, livestreamed	T-28	3w
LCH-05	LCH-05.03	Publish genesis file (embedding root,	Ops	Immutable IPFS + Arweave pin, hash	T-14	1w

		validator set, token allocations) 14 days before launch		published on Twitter + hln.net		
LCH -05	LCH -05. 04	Validator onboarding portal — KYC-free, stake + TEE attestation only	Ops + Frontend	≥ 25 000 independent validators ready by genesis	T-30 → T-8	8w
LCH -06	LCH -06. 01	**Mainnet Genesis Day** — coordinated start at UTC 14:00	Foundation + Core Tech	Block 0 produced, embedding root matches genesis file, livestream + countdown page	T=0	1d
LCH -06	LCH -06. 02	Day 0–7 war room — 24/7 coverage, hotfixes ready (pre-approved emergency governance)	All teams on standby	No rollback needed in first 7 days (target)	T+1w	1w

LCH-06	LCH-06.03	Public launch announcements, exchanges listings, major dApps go live	Marketing + Partnerships	≥ 15 dApps live, ≥ 3 Tier-1 CEX listings day-0	T=0	12w (parallel)
LCH-06	LCH-06.04	Post-launch audit & transparency report (first 30 days)	Security + Foundation	Published at T+30	T+30	4w

Cross-Cutting Final Safeguards

Task ID	Task	Deliverable	Timing
FINAL-01	Independent third-party genesis verification (multiple teams)	Signed letters confirming genesis file integrity	T-7 days
FINAL-02	Emergency multi-sig with 9-of-15 council able to pause rewards only (cannot touch funds)	Deployed & tested	T-30 days
FINAL-03	Full disaster recovery test (restore from genesis + Arweave snapshots)	< 4 hour recovery time	T-20 days

FINAL-04	Legal safe-harbour opinions (US, EU, Singapore, UAE)	Publicly posted	T-40 days

Detailed Sub-tasks:

Epic 1 – Enclave-Bound Anonymous Ingress & Mixer Network

Task ID	Task Description	Owner	Sub-tasks
ING-01.01	Design onion transaction format v1 (5-layer, Dilithium-signed outer, ML-KEM ephemeral keys per hop)	Crypto Lead	1. Research optimal onion layer structure (5 hops, Sphinx-like vs custom). 2. Define Protobuf schema for outer envelope and each layer. 3. Specify ML-KEM key encapsulation per hop. 4. Define outer Dilithium signature over entire onion. 5. Write serialization/deserialization spec. 6. Create test vectors for valid/invalid onions. 7. Review with security team for forward secrecy. 8. Finalize schema v1 and freeze.

ING-01.02	Implement wallet-side onion builder (random relay selection from DHT, layered encryption)	Wallet Team	1. Integrate DHT client for relay discovery. 2. Implement relay scoring (latency, uptime, attestation age). 3. Randomly select 5 distinct relays with diversity constraints. 4. Generate ephemeral ML-KEM keypairs per hop. 5. Perform layered encryption (outer → inner). 6. Sign outer layer with user Dilithium key. 7. Serialize and enforce <3.8 KB limit. 8. Add unit tests for malformed relay lists. 9. Integration test end-to-end onion creation.
ING-01.03	Add transaction size hard cap 3.8 KB serialized (post-onion)	Protocol Lead	1. Define consensus rule constant MAX_ONION_TX_SIZE = 3888 bytes. 2. Add check in mempool acceptance logic. 3. Add check in P2P message handler. 4. Write rejection reason code for oversized tx. 5. Generate 10 test vectors near boundary. 6. Update documentation and RPC error messages. 7. Add CI test that enforces limit.

ING-02.01	Create SGX/SEV/TrustZone relay enclave binary skeleton	TEE Team	1. Choose enclave framework (Fortanix EDP or Graphene). 2. Set up multi-platform build (SGX, SEV-SNP, CCA). 3. Create minimal enclave with edger8r interface. 4. Implement measurement reporting ecall. 5. Add host → enclave sealed storage stub. 6. Produce reproducible build pipeline. 7. Verify measurement hashes match across platforms. 8. Document enclave entry points.
ING-02.02	Implement process_onion_layer inside enclave	TEE Engineer	1. Implement ecall process_onion_layer(in_blob, out_blob, next_hop). 2. Decrypt own layer with enclave-static ML-KEM private key. 3. Validate layer format and hop count. 4. Generate forward blob for next hop. 5. Create remote attestation report. 6. Constant-time operations throughout. 7. Memory wipe after processing. 8. Unit tests via host simulation mode. 9. Performance benchmark (<10 ms per layer).

ING-02.03	Add remote attestation verification library	TEE Engineer	1. Integrate Intel IAS, AMD VCEK, and ARM DCAA verification. 2. Implement quote parsing and signature verification. 3. Cache valid reports with TTL. 4. Maintain CRL/revocation checking. 5. Reject debug-mode or outdated enclaves. 6. Add unit tests with known-good/bad quotes. 7. CI integration for attestation checks.
ING-02.04	Implement zero-logging guarantee	Security Lead	1. Audit all syslog, stderr, file writes in enclave/host. 2. Disable all logging inside enclave. 3. Implement mprotect + memset_s on all sensitive buffers. 4. Add host-side panic handler that wipes RAM. 5. Run Valgrind/memcheck on host. 6. Custom scanner for plaintext in core dumps. 7. Document no-logging policy in threat model.
ING-02.05	Add chaff/cover traffic generator inside enclave	TEE Engineer	1. Implement configurable chaff ratio (1–10×). 2. Generate dummy onion packets with random padding. 3. Use hardware RNG (RDRAND/DRNG) for content. 4. Schedule chaff with same jitter as real traffic. 5. Add admin ecall to adjust ratio live.

			6. Test traffic indistinguishability.
ING-02.06	Write side-channel mitigation checklist & constant-time crypto	Crypto Lead	1. Audit Dilithium and ML-KEM for branches/tables. 2. Replace with constant-time implementations. 3. Run dudect statistical tests. 4. Run ct-verif or FlowTracker. 5. Mitigate timing via fixed delays where needed. 6. Document all mitigations. 7. Pass power-analysis simulation suite.
ING-03.01	Final relay decrypts inner payload → Halo2 ZK transaction	TEE Engineer	1. Decrypt final layer inside enclave. 2. Verify inner payload format (Halo2 proof + tx). 3. Run Halo2 verification circuit. 4. Reject invalid or replayed tx. 5. Forward only verified tx to mempool queue.
ING-03.02	Final relay adds own attestation + monotonic secure timestamp	TEE Engineer	1. Source timestamp from enclave monotonic counter. 2. Sync counter periodically with attested NTP. 3. Include timestamp in forwarded message. 4. Sign with enclave Dilithium key.
ING-03.03	Final relay pushes to local shard’s mempool via encrypted Unix socket	Node Team	1. Set up Unix domain socket with SO_PASSCREDENTIALS. 2. Encrypt channel with per-session ML-KEM. 3. Host verifies enclave identity before accept.

			4. No plaintext in host memory.
ING-03.04	Reject double-spend attempts inside enclave before forwarding	Node Team	1. Maintain in-enclave seen-tx hash set (sealed storage). 2. Check tx hash against set. 3. On duplicate, drop and emit slashing evidence. 4. Rotate set every epoch.
ING-04.01	Implement Kademlia DHT extension for attested relays	P2P Team	1. Extend Kademlia key space with measurement hash. 2. Store (IP:port, attestation_report, expiry). 3. find_node filters invalid/expired attestations. 4. Periodic republish of own attestation. 5. Bootstrap from hard-coded seeds.
ING-04.02	Add DHT bootstrap nodes (10 geographically diverse)	Infra Team	1. Select 10 reliable hosted nodes across continents. 2. Generate long-term keys and attestations. 3. Hard-code into genesis config. 4. Document bootstrap process.
ING-04.03	Relay registry smart-contract that stakes 10k tokens to advertise	Protocol Lead	1. Design Move/Rust contract for relay registration. 2. Require stake + valid attestation proof. 3. Slashing conditions for fake attestation. 4. On-chain query interface for DHT.

ING-04.04	Wallet periodically refreshes relay list and prefers low-latency ones	Wallet Team	1. Background task every 10 min to query DHT. 2. Ping top 50 relays for RTT. 3. Score = latency × attestation freshness. 4. Cache top 100 relays locally.
ING-05.01	Implement AI cover-traffic scheduler (LSTM) inside enclave	ML Engineer	1. Train small LSTM on real vs chaff patterns. 2. Quantise to int8, <2 MB. 3. Port inference to enclave (Rust + micro-torch). 4. Dynamically adjust chaff ratio and jitter. 5. A/B test anonymity improvement.
ING-05.02	Add configurable padding to make all packets 1500 bytes	TEE Engineer	1. Pad all outbound packets to exactly 1500 bytes. 2. Fill with hardware RNG bytes. 3. Configurable via admin interface. 4. Test MTU compliance.
ING-05.03	Implement inter-packet delay jitter engine	TEE Engineer	1. Sample delay from normal distribution ±200 ms. 2. Apply per-packet before send. 3. Configurable mean and sigma. 4. Test timing fingerprint resistance.
ING-06.01	Build unlinkability auditor tool that replays full 5-hop paths	Security Researcher	1. Build libp2p simulator with 10k nodes. 2. Replay captured traffic with attestations. 3. Compute k-anonymity metrics. 4. Target k ≥ 1,000,000. 5. Generate report graphs.

ING-06.02	Run 72-hour traffic capture on testnet and prove no correlation	Red Team	1. Deploy traffic monitors on testnet relays. 2. Capture 72 h of real + chaff traffic. 3. Statistical correlation tests (Pearson, etc.). 4. Publish report with graphs.
ING-06.03	Publish formal anonymity proof using ProVerif or Tamarin	Academic Partner	1. Model 5-hop protocol in ProVerif. 2. Prove observational equivalence. 3. Write academic paper. 4. Submit to USENIX Security or PETS.

Cross-Cutting Tasks – Epic 1

Task ID	Task Description	Owner	Sub-tasks
X-01	CI pipeline for enclave builds with reproducible measurement hashes	TEE + DevOps	1. Docker-based reproducible builds. 2. Extract and assert measurement hashes. 3. GitHub Actions matrix for SGX/SEV/CCA. 4. Publish hashes as artefacts.
X-02	Fuzzing suite for onion parser and enclave entry points	QA + Security	1. libFuzzer harness for onion deserialization. 2. AFL++ for host components. 3. Run 48 h continuous. 4. Achieve 99% coverage. 5. No crashes found.

X-03	Threat model document (STRIDE) + attack tree	Security Lead	1. Run STRIDE workshop. 2. Build attack tree for mixer. 3. Document mitigations. 4. Publish on Notion/Miro.
X-04	External TEE security audit focused on Epic 1	Security Lead	1. Scope definition. 2. Select auditor (Trail of Bits). 3. Provide source + binaries. 4. Fix findings. 5. Receive final clean report.

Epic 2 – Verifiable Delay Witnesses & Blind Validation

Task ID	Task Description	Owner	Sub-tasks
BV-01.01	Design Verifiable Delay Witness (VDW) v1 format (Protobuf + fixed layout)	Protocol Lead	1. Define fixed-field layout for efficiency. 2. Specify Protobuf schema with version field. 3. Include tx_hash, shard_id, height, inclusion_proof, embedding_root, tee_sig. 4. Set max size 1800 bytes. 5. Create serialization examples. 6. Define forward-compatibility rules. 7. Review with ZK team for proof embedding. 8. Freeze schema v1.

BV-01.02	Implement VDW generation inside every shard node's TEE after embedding commitment	TEE Engineer	1. Hook into consensus finality event. 2. Collect required data inside TEE. 3. Generate inclusion proof (Halo2). 4. Sign with enclave Dilithium key. 5. Include fresh attestation report. 6. Serialize to blob. 7. Seal and hand to host for storage/serving. 8. Benchmark generation time (<200 ms).
BV-01.03	Add inclusion proof from tx → embedding_delta using Halo2 recursive circuit	ZK Engineer	1. Design recursive circuit for delta inclusion. 2. Implement in Halo2 with custom gates if needed. 3. Optimize for <750 bytes proof. 4. Benchmark verification on desktop and mobile. 5. Generate test vectors. 6. Add aggregation layer stub. 7. Achieve <45 ms desktop verify.
BV-01.04	Bundle VDW = {tx_hash, shard_id, height, inclusion_proof, embedding_root, TEE attestation + signature}	TEE Engineer	1. Define final binary format (version + concat). 2. Add length prefixes for parsing. 3. Implement bundling ecall. 4. Version field for future extensions. 5. Unit tests for parsing valid/invalid bundles.

			6. Documentation of fields.
BV-01.05	Persist every VDW for 5 years in shard-local encrypted DB	Node Team	1. Choose encrypted DB (SQLite with SQLCipher or RocksDB + AES-GCM). 2. Derive encryption key from enclave sealed storage. 3. Index by tx_hash and height. 4. Implement automatic pruning after 5 years. 5. Add Merkle history for post-prune proofs. 6. Backup strategy integration.
BV-02.01	Port Halo2 verifier to iOS (Swift + Rust FFI) and Android (Kotlin + JNI)	Mobile ZK Team	1. Build Halo2 as static lib for arm64/x86_64. 2. Create Swift FFI wrapper. 3. Create Kotlin JNI wrapper. 4. Benchmark on iPhone 15 and Pixel 9. 5. Optimize for <80 ms total. 6. Add error handling for malformed proofs. 7. Integration tests in Xcode/Android Studio.
BV-02.02	Implement WebAssembly + wasm-gc build of the same verifier	Frontend ZK Team	1. Compile Halo2 verifier to Wasm with wasm-gc. 2. Minimize binary size (<1 MB). 3. JS wrapper for async verification. 4. Benchmark in Chrome on mid-tier hardware. 5. Achieve <70 ms.

			6. Fallback to web workers.
BV-02.03	Build standalone “vdw-verify” CLI tool	Wallet Team	1. Rust binary with clap args (file/QR/base64 input). 2. Output colored VALID/INVALID + details. 3. Support all input formats. 4. Cross-platform builds. 5. Release on GitHub.
BV-02.04	Benchmark suite across platforms	Performance Lead	1. Define benchmark harness for all targets. 2. Run on CI matrix (iOS sim, Android emulator, Web, Desktop). 3. Enforce <80 ms threshold in CI. 4. Grafana dashboard for trends. 5. Publish baseline numbers.
BV-02.05	Add optional TEE-accelerated verification path	Mobile TEE Team	1. Detect Secure Enclave/Titan M availability. 2. Offload verification to TEE if present. 3. Graceful fallback to software. 4. Benchmark improvement. 5. Attestation of TEE verifier optional.

BV-03.01	Implement HTTP/3 + QUIC endpoint /vdw/:tx_hash with range support	Node Team	1. Add actix-web or axum handler. 2. Support Range headers for partial download. 3. Cache headers for CDN. 4. Rate limiting per IP. 5. Integration with storage backend.
BV-03.02	Add Cloudflare/Arweave/IPFS gateway integration	Infra Team	1. Pin every VDW to Arweave within 30 s. 2. Upload to IPFS via public gateway. 3. Cloudflare worker redirect. 4. Verify permanent availability. 5. Fallback chain.
BV-03.03	Implement Merkle-ised historical VDW buckets	Node Team	1. Bucket VDWs by day. 2. Build daily Merkle root. 3. Store roots on-chain or sealed. 4. Serve historical proofs with <50 KB data. 5. Prune local buckets after upload.
BV-03.04	Add rate-limiting + PoW challenge for public endpoints	Security Engineer	1. Implement adjustable PoW difficulty. 2. Cache solved challenges 60 s. 3. Per-IP rate limit fallback. 4. Monitor DoS attempts.

BV-04.01	Extend wallet to auto-fetch and cache VDW	Wallet Team	1. On tx submission, record tx_hash. 2. Poll or subscribe for VDW availability. 3. Store encrypted locally forever. 4. Background fetch on launch.
BV-04.02	Add “Export Proof” feature	Wallet Team	1. Generate QR code from VDW blob. 2. Save as .vdw file. 3. Copy base64 to clipboard. 4. All work offline after fetch.
BV-04.03	Implement “Show Proof” screen with offline verification	Wallet Team	1. Load cached VDW. 2. Run local verifier. 3. Display green check + details. 4. “Valid forever” messaging.
BV-04.04	Add VDW sharing via AirDrop, Nearby Share, deep-link	Mobile Team	1. Register hln://proof/ scheme. 2. AirDrop/Nearby integration. 3. Deep-link opens wallet → verify instantly. 4. Cross-platform testing.
BV-05.01	Design reorg handling: old VDWs become invalid	Consensus Team	1. Define REORG_INVALID error code. 2. Track canonical chain height. 3. Invalidate VDWs below reorg depth. 4. Spec document.

BV-05.02	Implement VDW invalidation Merkle tree per shard	ZK Engineer	1. Maintain revocation Merkle tree. 2. Issue revocation proofs ≤1 KB. 3. Serve alongside valid VDWs. 4. Light client verification path.
BV-05.03	Add background service that pushes replacement VDWs	Node + Wallet Team	1. Detect reorg on node. 2. Generate new VDWs. 3. Push via encrypted channel/push notification. 4. Wallet UI update message.
BV-05.04	Write formal specification of VDW finality rules	Protocol Lead	1. Define no invalidation after 10 crypto confirmations. 2. EIP-style document. 3. Review with researchers. 4. Publish.

Cross-Cutting Tasks – Epic 2

Task ID	Task Description	Owner	Sub-tasks
X2-01	End-to-end integration test: send tx → receive VDW → verify offline	QA Team	1. Script full flow on devnet. 2. Test all platforms. 3. Automate in CI. 4. Must pass forever.

X2-02	Fuzzing + property-based testing of VDW parser/verifier	Security + ZK	1. Quickcheck/Hypothesis generators. 2. 100 M witnesses. 3. No false positives/negatives. 4. CI enforcement.
X2-03	Formal verification of inclusion proof circuit	Academic Partner	1. Model in SAPIC/ProVerif. 2. Prove soundness and hiding. 3. Report publication.
X2-04	External audit of VDW format and verification code	Security Lead	1. Scope Halo2 recursion. 2. Select Kudelski/Trail of Bits. 3. Fix findings. 4. Clean report.
X2-05	Benchmark and optimisation target enforcement in CI	Performance Lead	1. CI fails if >80 ms on any platform. 2. Weekly optimisation review.

Epic 3 – Neural State Embeddings & Homomorphic Updates

Task ID	Task Description	Owner	Sub-tasks

EMB-01.01	Finalise LatentLedger Halo2 circuit specification	ZK + ML Lead	1. Define exact field arithmetic. 2. Constraint budget ≤8.2 M. 3. Write LaTeX spec. 4. Review cycles with cryptographers. 5. Include encoder/decoder symmetry. 6. Define supported state size limits. 7. Publish PDF spec.
EMB-01.02	Implement encoder transformer in Rust + Halo2 custom gates	ZK-ML Engineer	1. Transcribe 24-layer transformer to arithmetic circuit. 2. Custom gates for GELU/Rotary. 3. Optimize constraint count. 4. Unit tests per layer. 5. Benchmark synthesis time. 6. Achieve full transcription.
EMB-01.03	Implement decoder transformer (symmetric)	ZK-ML Engineer	1. Mirror encoder architecture. 2. Round-trip tests with random states. 3. 10^8 trials → zero error. 4. Constraint optimization. 5. Integration with encoder.
EMB-01.04	Generate trusted setup for LatentLedger circuit	Ceremony Team	1. Organize MPC with 100+ participants. 2. Use Powers of Tau + circuit-specific phase 2. 3. Verify contributions. 4. Destroy toxic waste. 5. Pin parameters on IPFS.

EMB-01.05	Write soundness proof for encoder/decoder pair	Academic Cryptographer	1. Formalize knowledge-soundness. 2. Prove simulation-extractability. 3. Write paper. 4. Target Crypto/IACR. 5. Peer review.
EMB-02.01	Design homomorphic delta format	ZK Engineer	1. Define δ = embed(new) - embed(old). 2. Support balance ops up to 2^128. 3. Fixed 512-byte format. 4. Test vectors. 5. Spec document.
EMB-02.02	Implement HomomorphicUpdate Halo2 circuit	ZK Engineer	1. Circuit: prev_embedding + delta → new_embedding + proof. 2. Optimize to ≤380 bytes proof. 3. <25 ms verify. 4. Recursive ready. 5. Benchmark suite.
EMB-02.03	Prove homomorphism is complete for supported operations	ZK Engineer	1. Formal proof document. 2. 10^7 random test vectors. 3. Cover transfer/mint/burn. 4. Edge cases (overflow).
EMB-02.04	Add batched homomorphic updates (up to 256 txs)	ZK Engineer	1. Aggregate deltas before update. 2. Single proof for batch. 3. ~1.5 bytes per tx overhead. 4. Modify consensus to use batches.

EMB-03.01	Generate synthetic training dataset	Data Engineer	1. Simulate 500 M tx sequences. 2. Include payments, DeFi, NFTs. 3. Store in Parquet on S3. 4. Anonymize any real traces. 5. Validation split.
EMB-03.02	Train initial compressor model	ML Team	1. PyTorch + DeepSpeed on 64×H100. 2. Target <1e-9 loss. 3. Quantise int4 + Huffman. 4. <8.2 MB checkpoint. 5. Hold-out testing.
EMB-03.03	Convert trained model → Halo2 custom gates	ZK-ML Engineer	1. Transcribe weights to circuit constants. 2. No accuracy loss vs FP32. 3. Layer-by-layer verification. 4. Final constraint count check.
EMB-03.04	Implement model distillation loop	ML Engineer	1. Hook into federated updates. 2. Retrain every 30 days. 3. Enforce ≤8 MB forever. 4. On-chain version tracking.
EMB-03.05	Benchmark compression	Performance Lead	1. Measure 100 kB state → 512 bytes. 2. ≥195× ratio. 3. Grafana + CI enforcement. 4. Compare vs Merkle.

EMB-04.01	Design recursive inclusion proof	ZK Engineer	1. Tx → delta → embedding path. 2. Plonkish + Halo2 recursion. 3. Target ≤800 bytes. 4. Spec v1.
EMB-04.02	Implement EmbeddingInclusion circuit	ZK Engineer	1. Full circuit implementation. 2. <60 ms verify on iPhone 15. 3. Aggregation support. 4. Test vectors.
EMB-04.03	Add aggregation: 256 inclusion proofs → one	ZK Engineer	1. Recursive aggregation circuit. 2. For batched light client updates. 3. Benchmark savings.
EMB-04.04	Write Nova-style folding scheme fallback	ZK Researcher	1. Implement IVC/folding alternative. 2. <90 ms on mobile. 3. Optional path in verifier. 4. Comparison paper.
EMB-05.01	Implement periodic full-shard re-embedding inside TEE	TEE + Node Team	1. Trigger every 1000 txs or 10 s. 2. Run encoder on current trie. 3. Commit new root. 4. Seal state.
EMB-05.02	Implement embedding root commitment in consensus	Consensus Team	1. Include in neural voting messages. 2. 67% agreement required. 3. Finality hook.

EMB-05.03	Add emergency “slow path” fallback	Safety Engineer	1. Detect divergence >1e-6. 2. Switch to Merkle root for one epoch. 3. Alert governance. 4. Auto-recovery.
EMB-05.04	Implement on-chain model upgrade ceremony	Protocol Lead	1. 30-day delay + staking vote. 2. New encoder/decoder without breaking history. 3. Migration tests.

Cross-Cutting Tasks – Epic 3

Task ID	Task Description	Owner	Sub-tasks
X3-01	End-to-end golden test: 1 M txs → reconstruction match	QA + ZK Team	1. Full simulation pipeline. 2. Exact state match. 3. CI gate forever.
X3-02	Continuous constraint optimisation	ZK Team	1. Weekly constraint count check. 2. Gate specialisation. 3. Target ≤8.2 M.
X3-03	External audit of LatentLedger circuit	Security Lead	1. 12-week audit by Trail of Bits + academic. 2. Focus soundness/homomorphism. 3. Clean report.
X3-04	Formal verification of homomorphic property	Academic Partner	1. Coq/Lean proof. 2. Machine-checked.

			3. Publication.
X3-05	Red-team embedding poisoning exercises	Red Team	1. Attempt state collisions. 2. Report findings. 3. Add mitigations.
X3-06	Model quantisation + compression pipeline	ML Engineer	1. int4 + Huffman pipeline. 2. Final ≤7.8 MB. 3. No accuracy loss.

Epic 4 – AI-Native Optimistic Consensus

Task ID	Task Description	Owner	Sub-tasks
CON-01.01	Define neural vote message format (Protobuf)	Consensus Lead	1. Design compact Protobuf schema. 2. Include node_id, predicted_embedding_hash, partial_BLS_sig, reputation_score, TEE attestation. 3. Max size ≤256 bytes. 4. Version field. 5. Test serialization round-trip. 6. Generate examples. 7. Freeze v1.
CON-01.02	Implement on-device distilled inference model	ML + ZK Engineer	1. Distill large model to ≤2 MB. 2. Input: current embedding + batched deltas. 3. Output: exact 512-byte hash. 4. <8 ms on Ryzen 7950X/M2. 5. Quantise int8. 6. Test accuracy vs full model. 7. Export for enclave.

CON-01.03	Integrate prediction model into node TEE	TEE Engineer	1. Load weights at startup (sealed). 2. Inference ecall only. 3. Attestation includes model hash. 4. Prevent weight extraction. 5. Benchmark inside enclave.
CON-01.04	Nodes broadcast signed neural vote within 150 ms	Node Team	1. Trigger on valid batch receipt. 2. Run prediction in TEE. 3. Sign partial BLS. 4. Gossip via P2P flood. 5. Measure 99th-pct latency on testnet. 6. Optimize network stack.
CON-02.01	Implement weighted threshold aggregation	Consensus Engineer	1. Aggregate partial BLS sigs. 2. Weight = stake × reputation. 3. Threshold ≥67% of active weight. 4. Instant finality event. 5. Handle late votes gracefully.
CON-02.02	After 67% agreement, commit embedding root + full BLS signature	Consensus Engineer	1. Finalize BLS threshold sig. 2. Commit to lattice history. 3. Emit finality event. 4. Trigger VDW issuance. 5. Persist for reorg protection.
CON-02.03	Implement “fast-path” gossip acceleration	P2P Team	1. Gossip only hash + partial sigs. 2. Lazy fetch full tx batch. 3. Reduce bandwidth 40×. 4. Prioritize vote messages. 5. Test under congestion.
CON-02.04	Add configurable fast-finality threshold	Protocol Lead	1. On-chain parameter (default 67%). 2. Governance proposal type. 3. 14-day activation delay. 4. Allow raise to 80% in attacks.

CON-03.01	Design challenge phase protocol	Consensus Engineer	1. Challenge window 800 ms. 2. Bond 0.5% stake. 3. Challenge message format. 4. Slashing rules. 5. Spec document.
CON-03.02	Implement Monte-Carlo dispute resolution in TEE cluster	TEE + ML Engineer	1. Sample 10 000 random executions. 2. Run in distributed TEEs. 3. Resolve <650 ms. 4. Output winning root. 5. Attested result.
CON-03.03	Implement fraud proof generation	ZK Engineer	1. Losing side generates Halo2 fraud proof. 2. 2 s timeout or slash. 3. Slash 1–5% based on discrepancy. 4. On-chain verification.
CON-03.04	Add economic finality timer	Consensus Lead	1. After 1.8 s no challenge → irreversible. 2. Game-theoretic analysis. 3. Internal paper. 4. Parameter tuning.
CON-03.05	Simulate 10 000 attack scenarios	Red Team + Researcher	1. Script 33% malicious, latency, eclipse. 2. Measure liveness/safety. 3. Fix weaknesses. 4. Final report.
CON-04.01	Implement reputation oracle inside TEE	ML + TEE Engineer	1. Ingest gradients + vote honesty. 2. Update 256-bit score. 3. Encrypted persistence. 4. Every 10 min update. 5. Remote attestation.

CON-04.02	Effective voting weight = stake × reputation	Consensus Engineer	1. Multiplicative weighting. 2. Reputation <0.1 → weight 0. 3. Auto-exit low reputation. 4. Test edge cases.
CON-04.03	Add reputation recovery mechanism	Protocol Lead	1. Honest nodes +0.02/week. 2. Prevent permanent exile. 3. Configurable rate.
CON-04.04	Reputation slashing for equivocation	Security Engineer	1. Detect double-voting. 2. Immediate 50% burn. 3. 7-day jail. 4. Evidence submission.
CON-05.01	Deploy 500-node global testnet with chaos	Testnet Ops	1. 5 continents, real latency. 2. Inject faults continuously. 3. Run 30 days. 4. Monitor KPIs.
CON-05.02	Instrument and enforce finality KPIs	Performance + QA	1. Median 600 ms probabilistic. 2. 1.8 s crypto. 3. >99.999% correct under 33% byzantine. 4. Grafana + CI gate.
CON-05.03	Run long-running stress test	QA + Infra	1. 10 M txs at 200 k TPS. 2. Measure finality distribution. 3. Tuning report.
CON-05.04	Publish security & performance paper	Researcher	1. Target IEEE S&P/USENIX. 2. Detail AI consensus model. 3. Peer review before mainnet.

Cross-Cutting Tasks – Epic 4

Task ID	Task Description	Owner	Sub-tasks
X4-01	End-to-end golden path test	QA Team	1. Tx → vote → 67% → finality → VDW. 2. Automate in CI. 3. 100% pass forever.
X4-02	Formal BFT proof under partial synchrony + AI model	Researcher	1. Extend existing proofs. 2. Lean/Coq or game theory. 3. Internal + external review.
X4-03	External consensus audit (three firms)	Security Lead	1. Trail of Bits, Runtime Verification, Informal. 2. Full scope. 3. Clean reports.
X4-04	Fuzzing + fault injection for vote messages	Security Team	1. 100 M malformed messages. 2. No invalid finality. 3. CI suite.
X4-05	Chaos-monkey automation	DevOps	1. Netem + custom byzantine. 2. 24/7 on testnet. 3. Auto-healing checks.

Epic 5 – Dynamic Neural Sharding

Task ID	Task Description	Owner	Sub-tasks
SHD-01.01	Finalise shard load metrics definition	Consensus + ML Lead	1. Define TPS, gas/s, embedding size, cross-shard ratio, p95 latency. 2. Publish spec v1.

			3. Implement collectors.
SHD-01.02	Implement per-shard time-series collector	Observability Team	1. Prometheus exporter. 2. 1-second granularity. 3. Export to VictoriaMetrics. 4. Dashboard templates.
SHD-01.03	Train LSTM load predictor	ML Engineer	1. Input: 120 s of 12 metrics. 2. Output: overload prob in 15 s. 3. >95% accuracy. 4. <1.2 MB model. 5. <4 ms inference.
SHD-01.04	Deploy predictor inside every node's TEE	TEE + ML Engineer	1. Weekly federated update. 2. Attestation proves version. 3. Inference ecall.
SHD-01.05	Add predictive alert → split/merge proposal	Node Team	1. Bond 0.1% stake. 2. Slash false positives >5%. 3. Proposal message.
SHD-02.01	Design state embedding bisection algorithm	ZK + ML Engineer	1. Deterministic split using shard_id + height seed. 2. Lossless round-trip. 3. Spec + proofs.
SHD-02.02	Implement SplitProposal message	Consensus Engineer	1. Old → new A + B embeddings. 2. Require 67% stake sigs. 3. Gossip + validation.
SHD-02.03	Execute live split inside TEEs	TEE Engineer	1. Re-execute recent txs on children. 2. <4 s completion. 3. No downtime.

			4. Coordinated checkpoint.
SHD-02.04	Migrate in-flight txs and mempool	Node Team	1. Route txs to correct child. 2. No loss or reorder. 3. Mempool split.
SHD-02.05	Update DHT + relay routing tables on split	P2P Team	1. Advertise new shard_ids. 2. Light clients update in 2 rounds. 3. Relay registry update.
SHD-02.06	Add rollback protection for failed split	Consensus Engineer	1. If no quorum, revert root. 2. Test with 30% byzantine. 3. Safety checks.
SHD-03.01	Implement merge trigger	Node Team	1. <10 TPS for 10 min → proposal. 2. 67% from both shards. 3. Bonded.
SHD-03.02	Design merge embedding algorithm	ZK + ML Engineer	1. Reverse bisection. 2. Deterministic + lossless. 3. Same seed method.
SHD-03.03	Execute live merge inside TEEs	TEE Engineer	1. Coordinated checkpoint. 2. <6 s completion. 3. State combine.
SHD-03.04	Retire old shard_ids after inactivity	Node Team	1. Purge after 1000 blocks. 2. Free disk. 3. Archive embeddings.

SHD-04.01	Implement Reed–Solomon erasure coding	Distributed Systems Lead	1. k=5, m=2 (7 replicas). 2. Embeddings + recent 10k txs. 3. Survive 40% loss.
SHD-04.02	Build AI placement optimiser	ML + Infra Engineer	1. Genetic algorithm every 10 min. 2. Minimize cross-region latency. 3. Reduce finality 800→320 ms.
SHD-04.03	Implement repair protocol	Node Team	1. Detect missing chunk. 2. Reconstruct from 5 pieces. 3. Push to new node. 4. <8 s repair.
SHD-04.04	Add geo-aware placement constraints	Infra Team	1. Max 2 replicas same region. 2. Validator metadata on-chain. 3. Enforcement.
SHD-04.05	Test 500 simultaneous splits + 40% crash	Chaos Team	1. Full recovery <30 s. 2. Pass before mainnet. 3. Video proof.

Cross-Cutting Tasks – Epic 5

Task ID	Task Description	Owner	Sub-tasks
X5-01	End-to-end integration test: 100 → 800 → 200 shards	QA Team	1. 30 min under real traffic. 2. Metrics + video. 3. CI automation.

X5-02	Formal proof of liveness & safety under dynamic membership	Researcher	1. Extend BFT proofs. 2. Academic review. 3. Publication.
X5-03	Fuzzing + property tests for split/merge math	ZK + QA	1. 10^8 trials. 2. No collisions. 3. CI suite.
X5-04	External audit of sharding logic	Security Lead	1. Runtime Verification + another. 2. 10-week audit. 3. Clean report.
X5-05	Real-time visualisation dashboard	Frontend Team	1. 3D lattice view. 2. Live splits/merges. 3. Public explorer.
X5-06	Chaos engineering suite	DevOps	1. Random splits, kills, partitions. 2. 24/7 staging. 3. Auto-reports.

Epic 6 – Useful-Work Economy & Federated Learning

Task ID	Task Description	Owner	Sub-tasks

UW-01.01	Design encrypted gradient format	ML-Crypto Engineer	1. Define Protobuf with int8 quantised weights. 2. Max 256 KB. 3. COSE_Encrypt0 with Dilithium key. 4. Include metadata (model version, node_id). 5. Test vectors. 6. Serialization in Rust/PyTorch.
UW-01.02	Implement secure aggregation protocol in TEE cluster	TEE + Crypto Engineer	1. Additive homomorphic masking. 2. Verifiable secret sharing. 3. 10k nodes → aggregate <8 s. 4. Zero leakage proof. 5. Dropout handling. 6. Attested aggregation result.
UW-01.03	Node-side gradient generation hook	Node + ML Engineer	1. After 1000 txs or 15 s. 2. Local training on anonymised batch. 3. <900 ms on 16-core. 4. ≤240 KB gradient. 5. Encrypt and submit.
UW-01.04	Add privacy filter: DP-SGD	Privacy Engineer	1. Per-example clipping 1e-6. 2. Noise σ=0.5. 3. (ε,δ)-DP bounds. 4. Opacus verification. 5. Integration tests.
UW-01.05	Implement drop-out tolerance	TEE Engineer	1. Succeed with ≥70% gradients. 2. Late acceptance in next round.

			3. No blocking.
UW-02.01	Build parameter server TEE cluster	TEE Ops	1. 16–32 instances (SGX/SEV/CCA). 2. Remote attestation required. 3. Secure channels only. 4. Geo-distribution. 5. Failover setup.
UW-02.02	Implement model update pipeline	ML Engineer	1. Every 10 min: decrypt → average. 2. Test on hold-out. 3. Publish if Δloss >0.003. 4. IPFS + on-chain hash. 5. Versioning.
UW-02.03	Add model validation oracle	Consensus Team	1. 100 staked validators re-infer. 2. Match within 1e-6. 3. Prevent poison. 4. Slashing for mismatch.
UW-02.04	Automatic rollback to previous model	Safety Engineer	1. Trigger on validation fail or finality drop >20%. 2. Within 2 min. 3. Governance alert.
UW-03.01	Design on-chain model registry contract	Protocol + Move Engineer	1. Store IPFS CID + hash + metadata. 2. Immutable versions. 3. 7-day delay for upgrades. 4. Dilithium-signed.

UW-03.02	Implement staking-weighted voting for major upgrades	Governance Team	1. 67% total stake. 2. 30-day voting. 3. New architecture only.
UW-03.03	Add model upgrade ceremony with Halo2 proof	ZK + ML Engineer	1. Prove homomorphic preservation. 2. ≤2 MB proof. 3. <3 s verify. 4. On-chain submission.
UW-03.04	Publish every model version on Arweave + Filecoin	Infra Team	1. Permanent pinning. 2. 10-year guarantee. 3. Redundant uploads.
UW-04.01	Implement Shapley-value approximation	Incentives Economist	1. Last-Value + periodic TMA. 2. >98% accuracy vs exact. 3. 10k-node samples. 4. Efficient computation.
UW-04.02	Deploy on-chain micro-payments	Tokenomics + Node Team	1. 0.02–0.15 HLN per gradient. 2. Batched every 10 min. 3. From inflation pool. 4. Receipt proofs.
UW-04.03	Add anti-gaming measures: gradient similarity clustering	Security Engineer	1. Detect >95% identical. 2. Slash clones. 3. >99% Sybil detection. 4. Test scenarios.
UW-04.04	Implement data-oracle work rewards	Oracle Team	1. Price feeds, fraud models = 3× reward. 2. Extra from oracle fees.

			3. Verification hooks.
UW-04.05	Economic simulation	Economist + Simulator	1. 100k nodes, 5 years. 2. Inflation <4%/year. 3. Useful-work dominates. 4. Open-source sim.

Cross-Cutting Tasks – Epic 6

Task ID	Task Description	Owner	Sub-tasks
X6-01	End-to-end test: 50k nodes → new model weekly	QA + Testnet Ops	1. 90-day public testnet run. 2. Measurable weekly improvement. 3. Stability checks.
X6-02	Formal privacy proof of federated pipeline	Privacy Expert	1. (ε=1.2, δ=1e-8) per 30 days. 2. External audit. 3. Publication.
X6-03	External audits: aggregation + incentives	Security Lead	1. CrypTFlow2 team + LeastAuthority. 2. Clean reports. 3. Fix findings.
X6-04	Public bug bounty for gradient poisoning	Security Team	1. Up to $1M. 2. Ongoing program. 3. Immunefi integration.

X6-05	Real-time AI dashboard	Frontend Team	1. Model accuracy, leaderboards. 2. Privacy budget. 3. Public explorer page.
X6-06	Token emission schedule + treasury contract	Tokenomics Team	1. 10-year curve. 2. Audited code. 3. Useful-work allocation.

Epic 7 – Quantum-Resistant Cryptography Suite

Task ID	Task Description	Owner	Sub-tasks
QR-01.01	Replace all ECDSA/EdDSA with CRYSTALS-Dilithium 3	Crypto Lead	1. Define SignatureScheme trait. 2. Implement Dilithium-3 wrapper. 3. Level 3 params. 4. Drop-in replacement. 5. Benchmark sizes.
QR-01.02	Integrate official NIST Dilithium code	Crypto Engineer	1. Use crystals-dilithium crate. 2. C + assembly bindings. 3. Pass all NIST KATs. 4. Wycheproof tests. 5. CI enforcement.
QR-01.03	Optimise Dilithium-3 for x86-64/AVX2 and ARM Neon	Optimisation Engineer	1. <60 µs verify Ice Lake. 2. <90 µs M2. 3. Assembly tweaks. 4. Published benchmarks. 5. CI thresholds.

QR-01.04	Add constant-time hardened implementation	Security Engineer	1. No secret branches/tables. 2. dudect + ctgrind passes. 3. FlowTracker audit.
QR-01.05	Replace every critical signature with Dilithium-3	Node + Protocol Team	1. Headers, votes, txs, VDWs. 2. Remove secp256k1/ed25519 paths. 3. Backward compat layer temporary.
QR-02.01	Implement ML-KEM-768 for enclave/node key exchange	Crypto Engineer	1. Replace X25519. 2. Onion, QUIC, TEE channels. 3. liboqs integration.
QR-02.02	Hybrid post-quantum handshake	Crypto Lead	1. ML-KEM-768 + X25519. 2. Per RFC draft. 3. Forward secrecy until 2035.
QR-02.03	Port liboqs into all builds	Integration Engineer	1. Node, enclave, mobile. 2. pq_crypto flag. 3. Single source truth.
QR-02.04	Benchmark handshake latency impact	Performance Team	1. <+12 ms vs classical. 2. 4G/5G/satellite tests. 3. Report.
QR-03.01	Add SPHINCS+-SHA256-192s-robust as backup	Crypto Engineer	1. ~41 KB sig, <1.2 s verify. 2. Cold wallets/genesis only. 3. Integration.

QR-03.02	Implement “hedged signing” mode	Wallet Team	1. Dilithium + optional SPHINCS+. 2. User toggle “Quantum doomsday”. 3. High-value txs.
QR-03.03	Pre-generate 10 000 SPHINCS+ keypairs in HSMs	Ops + Security	1. Never leave HSM. 2. Publish pubs at genesis. 3. Recovery keys.
QR-04.01	Design cryptographic agility framework	Protocol Lead	1. CryptoVersion enum. 2. Tagged sigs/KEMs. 3. No hard fork adds.
QR-04.02	Implement on-chain Crypto Upgrade governance	Governance Team	1. 180-day vote + 90-day migration. 2. Example: Dilithium-5 later.
QR-04.03	Add backwards compatibility for old signatures	Node Team	1. Verify legacy 24 months. 2. Disable via governance year 3.
QR-04.04	Build wallet migration tooling	Wallet Team	1. One-click PQ upgrade. 2. Batched tx. 3. Offline capable.
QR-04.05	Publish cryptographic continuity plan 2025–2040	Crypto + Governance	1. Triggers (NIST std, qubit milestones). 2. Signed PDF. 3. On-chain copy.

Cross-Cutting Tasks – Epic 7

Task ID	Task Description	Owner	Sub-tasks
X7-01	Full NIST + Wycheproof test vectors	QA Team	1. All primitives. 2. 100% pass in CI forever.
X7-02	External cryptography audit	Security Lead	1. PQShield + Kudelski/QuSecure. 2. Clean reports. 3. 10 weeks.
X7-03	Formal verification of constant-time impls	Academic Partner	1. Jasmin/Fiat-Crypto. 2. Machine-checked proofs.
X7-04	HSM + secure enclave integration for private keys	Security Team	1. CloudHSM, YubiHSM, etc. 2. Validator production.
X7-05	Quantum threat monitoring dashboard	Security Team	1. Qubit counts, papers. 2. Governance alerts. 3. Public page.
X7-06	Emergency “Quantum Break” hard fork playbook	Foundation	1. Activate SPHINCS+ <72 h. 2. Tested on staging.

Epic 8 – Extreme Scalability Layer

Task ID	Task Description	Owner	Sub-tasks

SCL-01.01	Implement fully parallel shard execution engine	Runtime Team	1. Tokio async per shard. 2. Rayon thread pool for tx execution. 3. Single 64-core node >180k TPS local. 4. Benchmark harness. 5. CPU affinity tuning. 6. No cross-shard blocking.
SCL-01.02	Add per-shard memory isolation	Systems Engineer	1. Linux cgroups for memory/CPU. 2. Jemalloc arenas per shard. 3. OOM killer isolation. 4. Test one shard OOM → others survive.
SCL-01.03	Cross-shard messaging via asynchronous mailbox	Runtime Team	1. Zero-copy lock-free ring buffers. 2. Async await on completion. 3. ≤1.1 s 99th-pct finality. 4. Testnet measurement.
SCL-01.04	Automatic load-balancing of shards across CPU/NUMA	Performance Lead	1. Work-stealing scheduler. 2. 85–95% CPU utilisation on 128-core. 3. Dynamic migration. 4. Monitoring hooks.
SCL-02.01	Implement native account abstraction	Protocol + Wallet Team	1. EIP-4337-like from genesis. 2. Any token/sponsor pays gas. 3. UserOperation struct. 4. Validation rules.

SCL-02.02	Add paymaster marketplace contract	Smart Contract Team	1. Third parties compete. 2. On-chain bids. 3. >95% new-user coverage on testnet. 4. Slashing for non-payment.
SCL-02.03	Bundler service (MEV-resistant)	Bundler Team	1. Aggregate 5k user-ops/sec. 2. Pay gas in HLN, rebate any ERC-20. 3. First 90 days zero user gas. 4. Privacy-preserving bundling.
SCL-02.04	One-click “Gasless mode” in reference wallets	Wallet Team	1. Toggle in settings. 2. Cached paymaster sigs. 3. Offline fallback. 4. UX testing.
SCL-03.01	Build AI fee predictor microservice	ML Engineer	1. Same LSTM as sharding. 2. Exact fee 10 s ahead. 3. >99.9% accuracy. 4. Public API endpoint.
SCL-03.02	Integrate fee predictor into wallet UX	Wallet Team	1. Show exact cost pre-sign. 2. “Your tx will cost exactly 0.00007 HLN”. 3. Never overpay. 4. Error handling.
SCL-03.03	Eliminate MEV completely	Consensus + Protocol	1. Priority = exact fee only. 2. No profitable reordering.

			3. 100k simulated txs → extractor profit 0.
SCL-03.04	Add “Fee smoothing” treasury	Tokenomics Team	1. Refund overpays on decongestion. 2. Automatic micro-refunds <10 min. 3. Treasury contract.
SCL-04.01	Enable recursive embedding compression for proofs	ZK Engineer	1. All proofs (VDW, inclusion, fraud). 2. 900× smaller than Polygon zkEVM. 3. Private transfer ~420 bytes. 4. Benchmark suite.
SCL-04.02	Implement Nova-style folding for mobile	ZK-Mobile Engineer	1. IVC fallback. 2. <60 ms on iPhone 15. 3. 10× compression. 4. Optional verifier path.
SCL-04.03	Add proof aggregation nodes	ZK Infra Team	1. Fold 10k proofs → one. 2. Daily sync 800 MB → 80 KB. 3. Specialised validators. 4. Incentives.
SCL-04.04	Benchmark end-to-end private payment	Performance + Mobile	1. Fresh device <1.2 s validation. 2. <2 KB data. 3. Public leaderboard. 4. CI enforcement.

Cross-Cutting Tasks – Epic 8

Task ID	Task Description	Owner	Sub-tasks
X8-01	1M+ TPS stress test (7 days)	QA + Infra	1. Real DeFi/NFT/GameFi load. 2. 5 continents. 3. Video + explorer proof.
X8-02	Global benchmark suite	Performance Team	1. Various hardware >500k TPS/node. 2. Published table. 3. Open-source tool.
X8-03	External performance + correctness audit	Security Lead	1. Runtime Verification. 2. 8-week audit. 3. Clean report.
X8-04	Gasless onboarding campaign	Marketing	1. Pre-fund paymasters 100M HLN. 2. Target 10M wallets first 30 days.
X8-05	Real-time scalability dashboard	Frontend Team	1. Live TPS, shards, fees. 2. Public explorer page.
X8-06	Emergency “Throttle” governance parameter	Protocol Lead	1. Cap at 200k TPS if needed. 2. Tested + documented.

Epic 9 – Developer & User Experience Layer

Task ID	Task Description	Owner	Sub-tasks

DX-01.01	Release hln-sdk v1 for multiple languages	SDK Team	1. Rust, TS/JS, Python, Go, Swift, Kotlin. 2. Identical APIs. 3. npm/pip/crates.io releases. 4. 500 shared e2e tests. 5. Documentation.
DX-01.02	Built-in ZK & embedding generation in SDK	ZK + SDK Engineer	1. sendPrivateTransfer() auto-handles onion/VDW. 2. Zero extra code for privacy. 3. Test coverage.
DX-01.03	Auto-paymaster selection in SDK	SDK + Wallet Team	1. Pick cheapest live paymaster. 2. New users $0 gas forever. 3. Fallback logic.
DX-01.04	Generate OpenAPI + GraphQL for JSON-RPC	DevRel Engineer	1. OpenAPI 3.1 spec. 2. Postman collection. 3. GraphQL wrapper. 4. Interactive docs.
DX-02.01	Implement hln deploy CLI	Tooling Team	1. Single binary all OS. 2. Compile → prove → deploy. 3. <6 s to mainnet. 4. QR code output.
DX-02.02	Support Move → Halo2 private contracts	Move + ZK Engineer	1. #[private] works out-of-box. 2. Circuit pipeline. 3. Proof generation.
DX-02.03	VS Code extension	DevRel + IDE Team	1. Syntax, autocomplete, proof estimator. 2. “Deploy to HLN” button. 3. Target 50k downloads.

DX-02.04	One-click testnet faucet in CLI/web	DevRel Team	1. Instant 100 HLN + paymaster credit. 2. Rate limited. 3. UX polish.
DX-03.01	Release one-click light client for all platforms	Mobile + Web Team	1. iOS/Android/Chrome/Edge/Firefox/Safari. 2. <100 KB sync. 3. <8 s first open. 4. Forever offline after.
DX-03.02	Web light client as embeddable Wasm	Web Team	1. COOP/COEP headers. 2. Single <script> tag. 3. iframe safe.
DX-03.03	Deep-link + QR schemes	Mobile Team	1. hln://pay/ and hln://proof/. 2. Scan → instant tx/proof. 3. <800 ms open.
DX-03.04	Progressive Web App wallet	Mobile + Security	1. Biometric login. 2. iCloud/Keychain recovery. 3. No seed phrase shown.
DX-04.01	Implement on-chain AI auditor	ML + ZK Engineer	1. 8 MB transformer. 2. 95%+ vulnerability detection. 3. Deploy-time scan. 4. Explanations.
DX-04.02	Add “Fix this for me” button	DevRel + AI Team	1. AI suggests patch. 2. One-click approve. 3. 80%+ auto-fix common bugs.

DX-04.03	Publish vulnerability leaderboards	Security + DevRel	1. $5M bounty pool. 2. Top 100 auditors ranked. 3. Integration.
DX-04.04	Formal verification bridge	Formal Methods Team	1. One-click export to Certora/Act/KEVM. 2. Used by top DeFi teams.

Bonus User-Facing Stories + Cross-Cutting – Epic 9

Task ID	Task Description	Owner	Sub-tasks
DX-05	“Send me $10 privately” natural UX	Wallet Team	1. Recipient scans QR/link. 2. No sender address visible. 3. All reference wallets.
DX-06	Private ENS (.hln names)	Protocol Team	1. Shielded resolution. 2. Only owner sees link. 3. vitalik.hln example.
DX-07	Privacy-preserving bridge UX	Wallet Team	1. ETH/Solana → private HLN. 2. One-click <15 s. 3. No address reuse.
DX-08	Mobile push for private incoming payments	Mobile Team	1. “You received money” only. 2. Reveal after biometric. 3. No amount/sender leak.

X9-01	100% e2e test coverage across SDKs/platforms	QA Team	1. CI breaks on any failure. 2. All languages.
X9-02	Public documentation site + playground	DevRel	1. Docusaurus. 2. Interactive examples. 3. 10k visits/week target.
X9-03	Hacker One / Immunefi bug bounty	Security	1. $10M pool. 2. Live at testnet.
X9-04	“Port to HLN” kit + grants for top dApps	Partnerships	1. Top 50 Ethereum/Solana. 2. $50k each. 3. 15+ live first 90 days.
X9-05	Global developer bootcamp tour	DevRel	1. 6 cities. 2. 5000+ trained. 3. Recordings online.

Epic 10 – Testnet → Mainnet Launch Sequence

Task ID	Task Description	Owner	Sub-tasks
LCH-01.01	Spin up closed Internal Devnet-Alpha	Core Tech + Ops	1. 100–200 team nodes. 2. All epics integrated. 3. 30-day run ≥300k TPS. 4. Zero crashes.
LCH-01.02	Run 168-hour chaos campaign	Chaos + QA	1. Random kills, 50% byzantine. 2. Full partition healing. 3. Self-heal SLA.

			4. Published report.
LCH-01.03	Freeze feature development	CTO	1. Tag v1.0.0-rc1. 2. No new features merged. 3. Branch freeze.
LCH-02.01	Launch Public Testnet-Omega with incentives	Testnet Ops + Token	1. 10M HLN faucet + rewards. 2. ≥10k nodes week 1. 3. ≥100k DAW by week 8.
LCH-02.02	Run staged load & chaos weeks	Community + Ops	1. 500k → 1M → 1.5M TPS. 2. Real dApps. 3. No rollback. 4. Leaderboard.
LCH-02.03	Incentive program for validators	Tokenomics	1. Top 100 5–20× multiplier. 2. Geographic/hardware diversity. 3. No >8% one region.
LCH-02.04	Genesis Rehearsal on testnet	Foundation + Core	1. Full dry-run. 2. Livestreamed. 3. 100% success.
LCH-03.01	Commission four parallel security audits	Security Lead	1. Trail of Bits, Kudelski, Runtime Verification, Academic. 2. Full scope. 3. Clean or low only.
LCH-03.02	Fix findings → re-audit until clean	Core Tech	1. Public commits. 2. Final reports.

LCH-03.03	Publish all four final audit reports	Foundation	1. On hln.net/security. 2. Attestation letters.
LCH-04.01	Launch $10M+ bug bounty	Security + Foundation	1. Immunefi critical up to $5M. 2. Quantum-break $10M. 3. 500+ whitehats.
LCH-04.02	Run red-team penetration test	External Red Team	1. Full insider access. 2. Clean or cosmetic report.
LCH-04.03	Pay out valid critical bounties	Foundation	1. Public transparency. 2. Zero unresolved at genesis.
LCH-05.01	Final tokenomics & emission schedule	Tokenomics + Gov	1. 10-year curve. 2. Useful-work treasury. 3. Signed PDF + on-chain.
LCH-05.02	Genesis allocation ceremony	Foundation + Legal	1. Multi-sig + TEE + HSM. 2. 4096 PQ key shares. 3. Livestreamed.
LCH-05.03	Publish genesis file 14 days before launch	Ops	1. IPFS + Arweave. 2. Hash on Twitter + site.
LCH-05.04	Validator onboarding portal	Ops + Frontend	1. KYC-free, stake + TEE only. 2. ≥25k validators ready.
LCH-06.01	Mainnet Genesis Day	Foundation + Core	1. UTC 14:00 start. 2. Block 0 produced. 3. Livestream + countdown.

LCH-06.02	Day 0–7 war room	All Teams	1. 24/7 coverage. 2. Hotfixes ready. 3. No rollback target.
LCH-06.03	Public launch announcements	Marketing + Partners	1. Exchanges, dApps live. 2. ≥15 dApps, ≥3 Tier-1 CEX day-0.
LCH-06.04	Post-launch 30-day audit & transparency report	Security + Foundation	1. Published T+30. 2. Metrics + incidents.

Cross-Cutting Final Safeguards – Epic 10

Task ID	Task Description	Owner	Sub-tasks
FINAL-01	Independent third-party genesis verification	Multiple Teams	1. Signed letters. 2. Confirm integrity T-7 days.
FINAL-02	Emergency multi-sig (pause rewards only)	Foundation	1. 9-of-15 council. 2. Deployed + tested T-30.
FINAL-03	Full disaster recovery test	Ops	1. Restore from genesis + snapshots. 2. <4 hour RTO.
FINAL-04	Legal safe-harbour opinions	Legal	1. US, EU, Singapore, UAE. 2. Publicly posted T-40.