KORTANA

Blockchain Technical Whitepaper

Version 2.0.0 • Production Ready

5s	<2s	50	1B
Block Time	Finality	Validators	Total Supply (DNR)

A production-grade Layer 1 blockchain engineered from the ground up for industrial-scale decentralized applications and global credit markets. Built entirely in **Rust** with **Delegated Proof-of-History (DPoH)** consensus.

Abstract

Kortana is a production-grade Layer 1 blockchain engineered from the ground up for industrial-scale decentralized applications and global credit markets. Written entirely in **Rust**, Kortana introduces a novel **Delegated Proof-of-History (DPoH)** consensus mechanism that fuses cryptographic time-ordering with Byzantine fault-tolerant finality, achieving **5-second block times** with **sub-2-second irreversible finality**.

Unlike conventional Layer 1 chains that force developers to choose between EVM compatibility and raw performance, Kortana provides a **Dual Virtual Machine** architecture: a full EVM execution engine for Solidity contracts alongside **Quorlin**, a custom stack-based VM optimized for high-throughput workloads. Both VMs share a unified state layer built on Merkle-Patricia Tries with SHA3-256 root hashing, ensuring cryptographic verifiability across execution environments.

The network is secured by 50 active validators operating under a delegated staking model with comprehensive slashing conditions — from 1% penalties for downtime to 100% stake burns for proven Byzantine behavior. The native **DINAR (DNR)** token features a deflationary economic model with a 50% base-fee burn mechanism and annual emission halvings.

1. System Architecture

Kortana's architecture is organized into five distinct, independently verifiable layers. Each layer is purpose-built and can be upgraded without affecting the others — a critical design choice for long-term maintainability of a production blockchain.

Application	JSON-RPC 2.0 compliant API supporting eth_chainId, eth_blockNumber, eth_gasPrice, net_version, and web3_clientVersion. Full compatibility with MetaMask, Remix, Hardhat, and all standard EVM tooling. WebSocket subscriptions for real-time block and transaction events.
Consensus	Delegated Proof-of-History provides a cryptographic clock via recursive SHA3-256 hashing, eliminating the need for inter-node time agreement. A Tower BFT finality layer runs atop the PoH sequence, achieving irreversible finality in under 2 seconds with 2/3 + 1 super-majority voting.
Execution	Dual VM engine: EVM with 50+ standard opcodes (arithmetic, stack, memory, control flow) and Quorlin with 25+ custom instructions (including local/global storage, event emission, and assertion primitives). Each transaction specifies its target VM, and both share a unified state trie.
State	Merkle-Patricia Trie with SHA3-256 root hashing. Each account stores nonce, balance, code hash, contract storage, and a contract flag. State snapshots enable fast node syncing without replaying the full chain history, while pruning keeps disk usage bounded.
Persistence	RocksDB tuned for high-throughput write amplification. Four storage domains: block store (indexed by height, hash, and slot), state snapshots (account history), receipt store (transaction execution results), and index DB (address → transaction history mapping).

Design Principles

Speed	5-second blocks and sub-2-second finality through deterministic leader scheduling on the PoH clock.
Security	Military-grade cryptography: SHA3-256, ECDSA signatures, BLS12-381 aggregation, and checksum-verified addresses.
Flexibility	Dual VM architecture lets developers choose between EVM compatibility and Quorlin's raw performance.
Decentralization	50 active validators with delegated staking, commission-based rewards, and democratic governance readiness.

2. Delegated Proof-of-History

The fundamental bottleneck in distributed systems is **time agreement**. Traditional blockchains like Bitcoin use Proof-of-Work as an implicit clock, while Ethereum's Proof-of-Stake relies on synchronized slot timers. Both approaches introduce latency because nodes must communicate to establish ordering.

Kortana eliminates this bottleneck with **Proof-of-History (PoH)** — a cryptographically verifiable passage-of-time mechanism. A designated leader continuously computes a recursive SHA3-256 hash chain, where each output becomes the input for the next hash. This creates an immutable, sequentially ordered record that proves time has elapsed between events without requiring any inter-node communication.

The PoH Hash Chain

```
// Recursive SHA3-256 hash chain — each hash proves elapsed time
hash[0] = SHA3-256(genesis_seed)
hash[1] = SHA3-256(hash[0])
hash[2] = SHA3-256(hash[1] || transaction_data)
hash[n] = SHA3-256(hash[n-1] || optional_data)
// Verification: replay N hashes to confirm sequence integrity
```

Transactions are woven into the hash chain by concatenating their data with the current hash before computing the next iteration. This creates an unforgeable timestamp that proves a transaction existed at a specific point in the sequence. Validators can independently verify sequence integrity by replaying the hash chain.

Slot-Based Block Production

- Each slot is exactly 5 seconds. The network produces one block per slot, yielding 17,280 blocks per day and approximately 6,307,200 blocks per year.
- An epoch consists of 432 slots (~36 minutes). Validator sets and leader schedules are recomputed at epoch boundaries based on current stake distributions.
- Leader election is deterministic and stake-weighted: validators with more delegated stake are selected more frequently, but every active validator gets proportional representation.
- If a leader misses their slot (due to downtime or network partition), the slot is skipped and the next leader in the schedule produces the following block. Missed slots count toward downtime slashing thresholds.

Byzantine Finality Layer (Tower BFT)

On top of the PoH clock, Kortana runs a PBFT-inspired finality protocol. After a block is proposed, validators cast **FinalizationVotes** that reference the PoH hash sequence number. When a block accumulates votes from validators controlling more than **2/3 + 1** of the total active stake, it becomes

irreversibly finalized. The finality delay is set to 20 slots, meaning blocks achieve absolute finality within approximately 1.5 to 2 seconds under normal network conditions.

The system includes **equivocation detection**: if a validator signs two conflicting votes for the same slot, the conflicting signatures serve as cryptographic proof of misbehavior, triggering automatic slashing of 33% of the validator's stake. Vote records are pruned after finalization to maintain constant memory usage.

3. Validators & Security

Kortana's security model is anchored by an economic incentive system that makes honest behavior profitable and Byzantine behavior catastrophically expensive. The validator set is capped at **50 active validators**, selected by total stake (self-bonded plus delegated).

Validator Lifecycle

- Registration: Validators must bond a minimum of 32 DNR and set a commission rate (0–100%, in basis points). The commission determines the validator's cut of delegator rewards.
- Delegation: Token holders can delegate their DNR to any registered validator, increasing that validator's total stake and probability of leader selection. Delegators earn proportional rewards minus the validator's commission.
- Active Set: At each epoch boundary, the top 50 validators by total stake are selected as the active set. Only active validators participate in block production and finality voting.
- Exit: Validators can voluntarily exit by unbonding their stake, subject to an unbonding period. Delegators can undelegate at any time with the same cooldown.

Slashing Conditions

Misbehavior is punished through automatic stake slashing. Slashing records are permanently stored on-chain, and repeated offenses compound the penalties. The four slashing conditions are:

Downtime **LOW • Penalty: -1%** Triggered after 50 consecutive missed blocks. The validator is automatically jailed for ~25 days (432,000 slots) and must manually unjail after the cooldown.	**Double Proposal** **MEDIUM • Penalty: -10%** Proposing two different blocks for the same slot. Detected by comparing block headers signed by the same validator key for identical slot numbers.
Equivocation **HIGH • Penalty: -33%** Casting conflicting finality votes for the same slot. Proven by presenting two valid signatures from the	**Byzantine** **CRITICAL • Penalty: -100%** Coordinated attacks such as censorship or state corruption. Results in full stake burn and permanent removal from the validator set. No unjailing possible.

same validator on contradictory FinalizationVote messages.	

Cryptographic Security

Hashing	SHA3-256
Signatures	ECDSA (secp256k1)
Aggregation	BLS12-381

4. Dual VM Execution Engine

Kortana solves the fundamental tradeoff between ecosystem compatibility and raw performance by implementing two virtual machines that share a unified state layer. Each transaction specifies its target VM type, and both VMs are gas-metered to prevent denial-of-service attacks.

EVM Execution Engine

A complete Ethereum Virtual Machine implementation supporting 50+ opcodes. Developers can deploy existing Solidity and Vyper contracts without modification. The EVM engine provides a full stack machine with 1024-depth stack, word-addressable memory, and persistent key-value storage.

Implemented Opcodes

Arithmetic: ADD, SUB, MUL, DIV, MOD, EXP

Stack: PUSH1–PUSH32, DUP1–DUP16, SWAP1–SWAP16, POP

Memory: MSTORE, MLOAD, MSTORE8

Control: JUMP, JUMPI, REVERT, RETURN, STOP

System: CALL, CREATE, SELFDESTRUCT, LOG0–LOG4

Tooling: Solidity • Vyper • Hardhat • Remix

Quorlin Virtual Machine

A custom stack-based VM purpose-built for high-throughput workloads. Quorlin provides 256 local variable slots, global key-value storage, and native event emission — all with lower gas costs than equivalent EVM operations, making it ideal for data-intensive applications.

Instruction Set

Arithmetic: Add, Sub, Mul, Div, Mod

Bitwise: And, Or, Xor, Not

Comparison: Eq, Lt, Gt

Storage: Load, Store, LoadGlobal, StoreGlobal

Control: Return, Emit, Revert, Assert, Dup

Features: Rust SDK • 25+ Opcodes • Low Gas • Events

Gas Metering

Both VMs use gas metering to bound computation. Each opcode has a defined gas cost, and transactions must specify a gas limit (minimum 21,000; maximum 10,000,000 per transaction). The block gas limit is capped at 30,000,000 to ensure block propagation stays within the 5-second slot window. Gas prices use a dynamic base fee adjusted per 2-block window, with a floor of 1 satoshi (10^{-18} DNR).

5. Token Economics

The **DINAR (DNR)** is the native utility token of the Kortana network, serving as the unit of account for gas fees, staking collateral, and governance weight. Its economic model is designed for long-term deflationary pressure while maintaining sufficient liquidity for network operations.

DNR	18	1B	5 DNR
Symbol	Decimals	Total Supply	Block Reward

Initial Distribution

Community & Ecosystem	60% — 600,000,000 DNR
Foundation Reserve	25% — 250,000,000 DNR
Team & Advisors	10% — 100,000,000 DNR
Developer Ecosystem Fund	5% — 50,000,000 DNR

Deflationary Mechanics

Base Fee Burn (50%)	Half of all transaction base fees are permanently burned, removing tokens from circulation. The other 50% goes to the block proposer as direct revenue.
Emission Halving (-10%/yr)	Block rewards decrease by 10% every 4,320,000 blocks (~1 year). Year 1 max emission: ~63M DNR. By year 10, block rewards fall below 2 DNR.
Slashing Burns (1–100%)	Slashed validator stake is burned, not redistributed. Byzantine attacks result in 100% stake destruction, providing an additional deflationary mechanism.

6. Extended Address Format

Kortana uses a unique **24-byte extended address format** that maintains full EVM compatibility while providing additional security through an embedded checksum. This design prevents address typos from resulting in lost funds — a problem that has caused millions of dollars in losses on other chains.

```
// Address structure: 24 bytes total
[ 20 bytes: EVM-compatible address ] [ 4 bytes: SHA3-256 checksum ]
// Derivation from public key:
address  = SHA3-256(public_key)[12..32]  // 20 bytes
checksum = SHA3-256(address)[0..4]       // 4 bytes
kortana_address = address || checksum     // 24 bytes
```

- The first 20 bytes are identical to a standard Ethereum address, ensuring full compatibility with existing wallets, contracts, and tooling.
- The trailing 4-byte checksum is verified on every transaction, rejecting addresses with typos before they reach the mempool.
- Contract addresses include a special flag byte in the checksum, allowing the network to distinguish between EOAs and contract accounts without a state lookup.
- Hex encoding supports both 0x-prefixed (48 chars) and 0x-prefixed + checksum (50 chars) formats, with automatic detection.

7. State Management

Kortana's state layer is built on a **Merkle-Patricia Trie** with SHA3-256 hashing, providing cryptographic proof of every account's state at any block height. The state root is included in every block header, enabling light clients to verify account balances without downloading the full chain.

Account Model

Every account, whether an externally owned account (EOA) or a smart contract, is represented by the same data structure in the state trie:

```
// Account state fields
address:    KortanaAddress  // 24-byte extended address
nonce:      u64             // Transaction counter (replay protection)
balance:    u128            // DNR balance in satoshi units
code_hash:  [u8; 32]        // SHA3-256 hash of contract bytecode
storage:    HashMap         // Contract key-value storage
is_contract: bool           // Contract flag
```

Mempool & Transaction Ordering

The mempool holds up to **10,000 pending transactions**, ordered by gas price in a priority queue. Transactions with identical gas prices are ordered by arrival time. The block selection algorithm greedily fills blocks up to the 30M gas limit, prioritizing higher-paying transactions. Transactions expire after 7 days (604,800 seconds) if not included in a block. Duplicate transactions are detected by hash and rejected at insertion time.

8. Network Infrastructure

Kortana's networking layer is built on **libp2p**, the same peer-to-peer framework used by Polkadot, Filecoin, and IPFS. This provides battle-tested primitives for peer discovery, encrypted communication, and efficient message propagation across a globally distributed validator set.

P2P Framework	libp2p
Encryption	Noise Protocol
Discovery	Kademlia DHT
Propagation	Gossipsub
Max Mempool	10,000 Tx
Active Validators	50 Nodes
Block Gas Limit	30,000,000
Storage Engine	RocksDB
Peer ID	32-byte Ed25519
NAT Traversal	AutoNAT + Relay
Tx Gas Limit	10,000,000 max
Min Gas/Tx	21,000

Persistence Architecture

All chain data is stored in a tuned RocksDB instance organized into four column families: the **block store** indexes blocks by height, hash, and slot for O(1) lookups; the **state store** persists account tries with snapshot support; the **receipt store** maps transaction hashes to execution results; and the **index store** maintains reverse mappings from addresses to their transaction history, enabling the explorer API to serve address pages without full-chain scans.