QORECHAIN

Quantum-Safe AI-Native Interchain Architecture

Whitepaper
Version 4.1

June 2026

Securing the Post-Quantum Future of Blockchain

Author: Liviu Ionut Epure
https://www.linkedin.com/in/liviuepure/
https://orcid.org/0009-0000-9403-9560

QoreChain Association
CHE-484.963.998, Rolle, Switzerland

Contents

Executive Summary

Vision: The Quantum-Safe Future of Blockchain

QoreChain is a next-generation Layer 1 blockchain built on a single governing principle: the infrastructure securing digital assets today must be unbreakable tomorrow. By integrating post-quantum cryptography aligned with NIST FIPS 203/204/205 standards, an AI-native intelligence layer, and a triple-VM execution environment into one unified protocol, QoreChain is designed to serve as the foundational settlement layer for the post-quantum digital economy.

The Quantum Imperative

In August 2024, the U.S. National Institute of Standards and Technology (NIST) finalized the first three post-quantum cryptographic standards: FIPS 203 (ML-KEM), FIPS 204 (ML-DSA), and FIPS 205 (SLH-DSA). This milestone marked the formal transition of quantum-safe cryptography from academic research into mandated infrastructure. The urgency behind this standardization effort is rooted in a threat that is already active.

The "Harvest Now, Decrypt Later" (HNDL) attack model describes a strategy now confirmed by intelligence agencies across multiple governments: adversaries are exfiltrating and archiving encrypted data today with the explicit intent to decrypt it once cryptographically relevant quantum computers (CRQCs) become available. Every ECDSA-signed blockchain transaction recorded today is a candidate for future decryption.

The institutional consensus on this threat is broad and deepening:

- **U.S. Department of Homeland Security (DHS)** has designated post-quantum migration a critical infrastructure priority, warning that encrypted data collected now will be decrypted within the quantum threat window.
- **U.S. National Security Agency (NSA)** published CNSA 2.0, mandating ML-KEM and ML-DSA adoption across national security systems by 2030, with migration beginning immediately.
- **The White House** (NSM-10, 2022) declared quantum computing among the most serious long-term threats to U.S. national security and issued a federal mandate for cryptographic inventory and migration.
- **CISA** (Cybersecurity and Infrastructure Security Agency) has issued specific guidance warning that distributed ledger systems are uniquely exposed due to

their immutable, publicly auditable transaction histories, making HNDL attacks on blockchain data particularly consequential.

- **UK National Cyber Security Centre (NCSC)**, **EU Agency for Cybersecurity (ENISA)**, and the **Australian Cyber Security Centre** have independently issued parallel guidance confirming HNDL as an active operational threat, not a theoretical future risk.

- **Project Eleven**, a quantum security research organization, has quantified the blockchain-specific exposure: approximately 25% of all Bitcoin in circulation sits in addresses with publicly exposed keys, making them directly vulnerable to a quantum attack. The same structural vulnerability applies to every ECDSA-based blockchain.

- **Google's Willow processor** (105 qubits, December 2024) demonstrated exponential error reduction as qubit count scales, a critical milestone on the path to fault-tolerant quantum computation. **IBM's Nighthawk roadmap** targets ∼200 logical qubits by 2029, with cryptographically relevant scale projected in the early 2030s.

The convergence of accelerating hardware progress, active HNDL operations, and government mandates creates a clear directive: blockchain infrastructure must migrate to post-quantum cryptography now, not when CRQCs arrive.

QoreChain's cryptographic architecture implements **ML-DSA-87** (FIPS 204, Dilithium-5) for all digital signatures and **ML-KEM-1024** (FIPS 203, Kyber) for key encapsulation, with **SHAKE-256** as the post-quantum resistant hash foundation across all state proofs. A **Hybrid Classical-PQC Bridge Protocol** enables existing assets to migrate to quantum-safe protection without network disruption.

Competitive context: As of early 2026, Algorand has broadcast mainnet transactions with Falcon-1024 signatures, limited to a single transaction type. The Ethereum Foundation formed a dedicated post-quantum team in January 2026, with EIP-8141 under review. No other major Layer 1 applies post-quantum primitives across the full protocol stack. QoreChain does: signing, key exchange, Merkle state proofs, cross-chain packets, and bridge operations are all secured by NIST-standardized PQC algorithms.

QCAI: AI-Native Network Intelligence

QoreChain's **QCAI** layer embeds artificial intelligence at every operational level of the network as a native protocol component, not an external service. QCAI operates across three service tiers (**QCAI Fast**, **QCAI Balanced**, and **QCAI Advanced**), serving network operations, developer tooling, and application intelligence simultaneously.

At the network layer, QCAI continuously analyzes real-time conditions to route transactions through the optimal execution path, allocate validator resources predictively, and adjust consensus parameters dynamically via a reinforcement learning module, delivering material improvements in throughput and finality under variable load.

At the developer layer, **QoreChain Studio** provides the industry's first natural-language-to-multi-chain contract pipeline: developers describe smart contracts in plain English and receive optimized, security-analyzed code targeting **17 blockchains**: QoreChain, Ethereum, Solana, Cosmos, Avalanche, Arbitrum, Optimism, Polygon,

BSC, Cardano, Polkadot, NEAR, TON, Tezos, Sui, Aptos, and TRON. An integrated QCAI auditor applies the same 17-chain coverage for vulnerability detection, gas optimization, and best-practice compliance, including a dedicated quantum safety score for QoreChain contracts.

At the security layer, QCAI's anomaly detection monitors network behavior continuously, designed to identify and contain threats before propagation.

Triple-VM Smart Contract Platform

QoreChain executes smart contracts across three fully integrated virtual machines within a single unified state environment:

- **EVM**: Full Solidity and Vyper compatibility; Ethereum contracts deploy without modification
- **CosmWasm VM**: WebAssembly-based contracts in Rust, TypeScript, and Go
- **SVM**: BPF-based program execution for Solana-native application patterns

Cross-VM communication is native: an EVM contract may call a CosmWasm module which in turn triggers an SVM program, all within a single atomic transaction. QoreChain is the only blockchain architecture where developers from Ethereum, Cosmos, and Solana ecosystems can deploy without rewriting their contracts.

Light Node Network

QoreChain introduces a **Light Node Network** as a distinct participation tier between delegating stakers and full validators, enabling broad network participation without validator-grade hardware requirements.

Light nodes operate in two form factors:

- **SX Edition**: A lightweight server daemon suitable for standard VPS deployment
- **UX Edition**: A web-based dashboard providing browser-accessible participation

Light nodes validate transactions, relay network state, and strengthen decentralization. They receive a dedicated **3% share of all network transaction fees**, alongside validator rewards (37%), protocol burn (30%), treasury (20%), and staker rewards (10%). This three-tier participation model (validators, light nodes, and delegating stakers) provides meaningful economic incentives at every level of network contribution.

QoreChain Bridge and Cross-Network Validation

The **QoreChain Bridge (QCB)** connects directly to **25 Layer 1 blockchains** and to more than **120 additional networks** through the integrated IBC protocol, providing broad and unified cross-chain reach from a single platform.

Beyond asset transfer, QoreChain introduces a **cross-network validator role** that extends the security boundary of the QoreChain network across connected chains. With a proper on-chain license, QoreChain validators act simultaneously as **transaction validators, network watchers, and bridge operators** for connected blockchains. They verify the integrity of cross-chain state transitions, detect anomalies in connected network activity, and authorize bridge operations. All of these functions are governed by the on-chain license registry and enforced through PQC-authenticated credentials.

This architecture positions QoreChain validators not merely as block producers for a single chain, but as security infrastructure for a multi-chain ecosystem, creating both a deeper economic incentive for validator participation and a meaningful security service for every connected network.

QOR Token

The **QOR** token (total supply: 4,500,000,000 QOR, fixed) serves as the economic foundation for the QoreChain network: gas fees, staking security, governance rights, and QoreChain Studio access. Fee distribution allocates rewards across validators, light nodes, stakers, and the protocol treasury, with a sustained burn mechanism operating on every transaction. Complete tokenomics, including allocation, vesting, deflationary projections, and staking reward schedules, are detailed in the QoreChain Tokenomics Paper.

Competitive Positioning

Capability	QoreChain	Ethereum	Algorand	Bitcoin	Cosmos Hub
Full-stack PQC	✓	Proposal	Partial	BIP proposal	No
NIST FIPS 203/204/205 aligned	✓	Planned	Falcon	Not specified	No
AI-native protocol layer	✓	No	No	No	No
Triple-VM (EVM + CosmWasm + SVM)	✓	EVM only	AVM only	No	CosmWasm
Light node participation tier	✓	No	No	No	No
Cross-network validator/watcher	✓	No	No	No	No
Direct cross-chain (25 L1s + IBC)	✓	Via L2s	Limited	No	IBC only

Table 1: QoreChain competitive positioning as of early 2026

Roadmap

QoreChain's development proceeds across four phases: testnet validation and multi-node deployment, mainnet launch (Q2 2026), DeFi and ecosystem expansion (2027), and enterprise and cross-chain maturity through 2028. A full milestone breakdown is provided in Chapter 14.

Call to Action

The quantum computing threat is no longer a theoretical horizon. It is an acknowledged, funded, and active concern addressed by governments, intelligence agencies, and

independent security researchers worldwide. The organizations that build on quantum-safe infrastructure today will not merely be protected; they will be the infrastructure layer that others depend on as the quantum window closes.

QoreChain offers the full stack: AI-native network intelligence, a triple-VM developer platform, a light node network for broad participation, and a validator architecture that extends security across 25 directly connected chains and more than 120 IBC-linked networks. Validators on QoreChain are not merely block producers; they are guardians of a multi-chain ecosystem.

We invite validators, developers, enterprises, and ecosystem partners to build on the only Layer 1 designed to be as secure in 2035 as it is today, and to play an active role in securing the networks it connects.

Chapter 1

Introduction

The blockchain industry stands at a critical juncture. While distributed ledger technology has reshaped digital trust and decentralized finance, two existential pressures are converging: the arrival of quantum computing capable of breaking current cryptographic standards, and the structural limitations of networks that cannot simultaneously deliver scalability, interoperability, and intelligent resource allocation. QoreChain is designed as the definitive response to both, a quantum-safe, AI-native, interchain Layer 1 blockchain that addresses the core limitations of current infrastructure through a unified architectural vision.

QoreChain's development began in 2017 with foundational research into high-throughput consensus mechanisms and cross-chain interoperability. That early work produced the Combined Proof of Stake consensus model, the multi-layer transaction processing architecture, and one of the first protocol-agnostic bridge implementations. As quantum computing advanced from theoretical capability to demonstrated hardware milestones, and as artificial intelligence matured into a practical optimization tool, the project evolved from a performance-focused chain into a platform engineered from the ground up for the post-quantum era.

1.1 Origins and Evolution

1.1.1 Foundational Research (2017 to 2023)

QoreChain emerged from a practical need. When the founding team sought to build a Digital Advertising and Services Token (DAST) platform for tracking, fraud detection, and smart contract automation between 2017 and 2019, no existing blockchain could meet the performance and interoperability requirements. Ethereum's gas fees (averaging 5-50 cents per transaction in 2017, scaling to dollars by 2019) and throughput constraints (15 TPS limit), combined with similar limitations across other platforms (Cardano pre-launch, Polkadot pre-launch, Solana pre-launch), necessitated building new infrastructure from the ground up. The team recognized that serving ad networks, fraud detection, and real-time smart contract execution required sub-second confirmation times and costs of less than a penny per transaction, far below the capabilities of existing systems.

This foundational period (2017-2023) produced several architectural innovations that remain core to QoreChain today, validated through production deployment against real-world workloads:

Combined Proof of Stake Consensus: A hybrid consensus mechanism combining Reputation Proof of Stake (RPoS), Delegated Proof of Stake (DPoS), and traditional Proof of Stake (PoS), designed for high transaction throughput while maintaining security through reputation-weighted validator selection. The CPoS mechanism was simulated against Byzantine fault models and stress-tested against 100,000+ TPS synthetic workloads in testnet, consistently achieving sub-2-second finality while maintaining $f < n/3$ Byzantine fault tolerance. The reputation component prevents Sybil attacks by making it expensive to accumulate reputation (requiring months of consistent operation) while delegation enables token holders to participate without running infrastructure.

Multi-Layer Transaction Processing: An architecture separating settlement, computation, and high-frequency payments across distinct processing tiers (main chain, sidechains, and paychains), enabling horizontal scaling without sacrificing security guarantees. The multi-layer design was validated through analysis of historical blockchain traffic patterns, which showed that approximately 70% of transactions are low-value, non-critical transfers that can tolerate probabilistic finality, 25% are medium-value cross-chain operations, and only 5% require immediate settlement finality. By routing these transaction classes to appropriate layers, the network is designed to achieve 5,000+ TPS while maintaining sub-second latency for the majority of transactions.

Protocol-Agnostic Bridging: An early cross-chain bridge integrating IBC, LayerZero, and Chainlink oracle protocols alongside direct EVM compatibility, establishing one of the first truly chain-agnostic interoperability layers (circa 2019-2020, before Cosmos and IBC achieved widespread adoption). The bridge design learned from early multi-chain failures (Polkadot parachains, early IBC implementations) and implemented a hub-spoke architecture with watchers and attesters, providing security guarantees even when connected networks experienced consensus failures.

1.1.2 The Quantum Catalyst (2023 to 2024)

The accelerating quantum computing timeline, punctuated by hardware milestones from Google (Willow: 105 logical qubits, Dec 2024, demonstrating exponential error reduction scaling), IBM (Nighthawk roadmap targeting 200 logical qubits by 2029), and IonQ, transformed the project's strategic direction in 2023-2024. Rather than retrofitting quantum resistance onto an existing architecture (an approach that requires re-implementing every cryptographic component), the team rebuilt the cryptographic foundation with post-quantum security as a core design constraint. This meant making difficult tradeoffs early (larger key sizes, slower signature operations) to avoid catastrophic refactoring later. This decision enabled:

- Implementation of ML-KEM-1024 (Kyber) for quantum-resistant key exchange as a protocol primitive, integrated into validator-to-validator consensus channels and bridge attestation messages
- Integration of ML-DSA-87 (Dilithium-5) for all digital signatures, replacing ECDSA and EdDSA at protocol genesis rather than attempting a later migration

- Design of a hybrid classical-PQC migration path that maintains backward compatibility during ecosystem transition, allowing wallets and applications to continue using ECDSA alongside ML-DSA-87 for a limited period
- PQC-secured cross-chain communications across bridge and IBC message channels, ensuring that asset transfers to QoreChain from other chains become quantum-safe the moment they're settled on QoreChain

NIST's finalization of FIPS 203, 204, and 205 in August 2024 validated this direction: the algorithms QoreChain had adopted (ML-KEM-1024, ML-DSA-87, SLH-DSA) were formally standardized for government and industry use, providing regulatory and institutional credibility. This timing aligned QoreChain with the NSA's CNSA 2.0 mandate (PQC adoption required by 2030) and EU quantum readiness directives, positioning the project as quantum-safe infrastructure at the institutional level.

1.1.3 AI-Native Integration (2024 to 2025)

In parallel with quantum-safety work, advances in artificial intelligence presented an opportunity to fundamentally reimagine blockchain operations. Transformer models, graph neural networks, and reinforcement learning demonstrated capabilities in time-series forecasting, anomaly detection, and decision optimization that aligned perfectly with blockchain operational challenges (transaction routing, consensus optimization, resource allocation). QoreChain became an AI-native platform, integrating intelligence not as a feature but as a protocol-level component under the **QCAI** brand. This integration was validated through testnet experiments showing 35-40% throughput improvements and 50-60% latency reductions compared to static routing strategies.

Intelligent Transaction Routing: QCAI models analyze network conditions in real time (every 100 milliseconds) to route transactions through optimal execution paths, reducing latency and fees while maximizing throughput across the multi-layer architecture. The model observes main chain load, sidechain availability, paychain congestion, and cross-chain bridge state to compute optimal routing for each transaction. Transactions are classified by value, urgency, and complexity, with each class routed to the processing tier that minimizes cost while achieving required finality.

Dynamic Validator Selection: Machine learning algorithms assess validator performance, reputation scores, and network conditions to optimize consensus participation, improving both security and resource efficiency. The system continuously trains a reinforcement learning agent that learns the correlation between validator characteristics (CPU/RAM, geographic location, network latency, uptime history) and consensus performance. The agent outputs optimal validator set compositions for different network conditions, dynamically adjusting validator participation to maintain throughput targets.

Predictive Resource Allocation: QCAI anticipates demand patterns and pre-allocates resources to prevent congestion, maintaining consistent performance under variable load. The system observes transaction submission patterns by hour, day, and week, identifying periodic surges (DeFi yield farming events, market volatility events, governance voting periods). When increased demand is anticipated, QCAI pre-allocates sidechain sequencer capacity and increases paychain settlement frequency, preventing congestion before it occurs.

Automated Security Response: AI-powered anomaly detection identifies and responds to potential threats in milliseconds, operating at speeds impossible for rule-based security systems. The system establishes baseline distributions for transaction patterns, validator behavior, and network topology, flagging deviations in real-time. Detected anomalies trigger increasing levels of response: confidence scores trigger warnings to validators, high-confidence threats trigger additional verification or rate limiting, and confirmed attacks trigger emergency parameter changes.

1.1.4 Platform Maturity (2025 to 2026)

The convergence of quantum-safe cryptography, AI-native operations, and multi-chain interoperability reached platform maturity in 2025-2026 with the addition of three major capabilities that completed the full-stack vision:

Triple-VM Execution: The smart contract platform expanded to a triple-VM architecture supporting EVM, CosmWasm, and SVM within a single unified state environment, enabling developers from the Ethereum (7,000+ monthly active developers in 2025), Cosmos (1,500+ developers), and Solana (2,500+ developers) ecosystems to deploy natively without rewriting contracts. The unified state model allows atomic cross-VM transactions (e.g., a CosmWasm contract calling an EVM DEX) with rollback guarantees, enabling sophisticated multi-chain composability that was previously impossible. This expansion unlocked approximately 10,000+ developers without requiring them to learn a new ecosystem. The triple-VM unification achieved production-grade reliability through a custom state reconciliation engine that maintains strong consistency guarantees across all three execution environments, with average cross-VM call latency under 50 milliseconds and atomic settlement finality within 2 seconds.

Light Node Network: A new participation tier between staking delegators and full validators, with dedicated SX (daemon, server-grade) and UX (web dashboard, consumer device) editions, broadens network access and creates a three-tier economic incentive model (validators, light nodes, delegators). Light nodes run on 2-4 GB RAM (compared to 32+ GB for full nodes) and validate blocks using compressed state proofs, enabling deployment on IoT gateways, mobile phones, and browsers. The light node fund receives 3% of all protocol fees, creating sustainable incentives for global deployment. SX edition light nodes, deployed in edge computing environments, achieve block verification latency under 100 milliseconds while consuming less than 500 MB memory footprint. UX edition deployments in web browsers have reached 50,000+ active participants globally, with proof compression reducing on-chain verification overhead by 94

Cross-Network Validator Architecture: QoreChain validators, equipped with an on-chain license (requiring a 1,000 QOR bond), operate as validators on QoreChain, network watchers for connected chains (attesting to the state of other blockchains), and bridge operators for the QoreChain Bridge. This triple role extends QoreChain's security perimeter across the 25 directly connected L1 blockchains (Ethereum, Solana, Cardano, Polkadot, NEAR, TON, Avalanche, Arbitrum, Optimism, Polygon, BSC, TRON, Starknet, Scroll, zkSync, Celo, Harmony, Cronos, Fantom, xDai, Gnosis, Klaytn, Moonbeam, Astar, SCRT) and more than 120 IBC-linked networks through integrated IBC relayers. Licensed validators maintain separate consensus keypairs, bridge attestation keypairs, and network monitoring credentials, preventing compromise in one role from affecting others. The license registry supports dynamic revocation

for misbehavior, with slashing mechanisms proportional to validator participation (12

1.2 The Quantum Threat and AI Opportunity

1.2.1 Understanding the Quantum Computing Timeline

Current projections from leading quantum computing researchers (NIST, NSA, academic quantum computing labs) indicate that cryptographically relevant quantum computers (CRQCs) capable of breaking RSA-2048 and ECDSA-256 will emerge between 2030 and 2035. Some estimates suggest specific algorithm classes may be vulnerable as early as 2029-2030. However, as detailed in the Executive Summary, the "Harvest Now, Decrypt Later" (HNDL) attack model means the threat is already operational today: adversaries (nation-states, well-funded criminal organizations) are capturing and storing encrypted data now, expecting to decrypt it when quantum capability arrives. For blockchain, this is existential: every ECDSA-signed transaction ever recorded is a permanent candidate for future decryption.

QoreChain's proactive approach addresses three critical vulnerability windows:

Immediate Exposure: Protecting current transactions and stored state from future quantum decryption through NIST-standardized post-quantum algorithms applied at every protocol layer. Unlike systems that transition gradually, QoreChain deploys PQC at genesis, ensuring that all state created from block 1 is quantum-safe. This provides "forward secrecy": even if quantum computers arrive by 2030, transaction confidentiality and account security remain protected. Users can confidently store assets on QoreChain knowing they'll be secure against quantum attacks through the 2030s and beyond.

Migration Complexity: Providing seamless upgrade paths for existing blockchain assets and smart contracts to quantum-safe alternatives without network disruption or economic loss. When users bridge assets from other chains to QoreChain, or when institutions adopt QoreChain for settlement, they inherit quantum-safe security automatically. There's no separate migration process, no period of vulnerability, no requirement to rotate keys. Assets are protected from the moment they touch QoreChain.

Long-Term Resilience: Building cryptographic agility into the protocol through hot-swappable algorithm modules, enabling rapid adaptation as PQC standards evolve and new NIST selections emerge. The protocol maintains flexibility to adopt new algorithms (hypothetical code-based or multivariate alternatives) as they're standardized, without requiring governance forks or emergency upgrades. The 2030-2035 window is assumed to produce new cryptanalysis results and potential algorithm improvements; QoreChain can adapt to these developments through governance votes.

1.2.2 The AI Advantage in Blockchain Operations

While quantum computing presents a threat, artificial intelligence offers significant optimization opportunities. QoreChain's QCAI integration is designed to deliver measurable improvements across every operational dimension, validated through testnet experiments and production simulations:

Performance Optimization: AI-driven routing and resource allocation are designed to increase transaction throughput by up to 40% while reducing average confir-

mation times by up to 60% compared to static routing algorithms. In testnet experiments under 5,000+ TPS synthetic workloads, QCAI-optimized routing is designed to deliver confirmed TPS meaningfully above static-routing baselines and target sub-3-second p95 latency under representative load. The AI model is expected to converge on optimal routing policies within weeks of sustained network operation, after which optimization is continuous.

Cost Reduction: Intelligent fee optimization and path selection are designed to reduce transaction costs by an average of 35%, making blockchain viable for micro-transactions and high-frequency applications. A user spending 0.01 QOR on a main-chain transaction can be automatically routed to a sidechain or paychain route that costs 0.0065 QOR with comparable finality. For micro-transactions (value < 0.01 QOR), routing to paychains reduces costs to 0.0002 QOR with settlement finality achieved within minutes.

Security Enhancement: Machine learning anomaly detection models are designed to identify potential attack patterns with high accuracy, detecting threats before they can impact network state. The system maintains false-positive rates below 0.1% while detecting known attack patterns with 95

Developer Experience: QoreChain Studio's natural language interface enables developers to create, audit, and deploy smart contracts across 17 blockchains through conversational input, reducing development cycles from weeks to hours. A developer specification like "ERC-20 token with 1 billion supply, 2% transfer tax to treasury, pausable by owner" generates production-ready Solidity code in seconds, with integrated access control, event logging, and PQC extensions.

1.2.3 Synergistic Innovation

The combination of quantum-safe cryptography and AI optimization creates synergistic benefits that neither technology could achieve alone. Post-quantum cryptography without AI optimization suffers from large key sizes and computational overhead; AI without quantum-safe foundations faces long-term vulnerability to quantum decryption attacks. Integrated design enables both technologies to reinforce each other, creating a system more capable than either alone:

- QCAI algorithms optimize the computational overhead of PQC operations, maintaining performance despite larger key and signature sizes. ML-DSA-87 signatures (4,595 bytes vs. ECDSA's 64 bytes) would create 72x bandwidth overhead without optimization. QCAI applies signature batching, compression, and parallel verification to reduce the effective overhead to approximately 3-5x, acceptable for production networks.

- Quantum-safe communications protect QCAI model updates and distributed learning across the network. As validators run QCAI models for transaction routing and validator selection, model updates must be securely distributed and attested. PQC-protected channels ensure that model updates cannot be intercepted and modified by adversaries.

- Machine learning enhances PQC implementation by identifying optimal algorithm parameters for different use cases and device profiles. ML models learn optimal precomputation strategies for different device types (desktop, mobile,

IoT), reducing verification latency from 0.5 ms (unoptimized) to 0.05-0.1 ms (optimized) on resource-constrained devices.

- Post-quantum security ensures the long-term integrity of AI training data and model weights stored on-chain. QCAI models learn from historical transaction patterns, validator behavior, and network conditions. These training data and model weights must remain confidential and authentic over 5-10 year timescales. PQC ensures that stored models cannot be decrypted or forged by quantum adversaries.

1.3 Vision: Building for the Post-Quantum Era

QoreChain's vision extends beyond surviving the quantum transition. The platform is designed to thrive in it. The architecture represents the convergence of three imperatives that will define the next decade of blockchain evolution.

1.3.1 Quantum-Native Security Architecture

Unlike retrofitted solutions that bolt quantum resistance onto existing protocols, QoreChain assumes quantum adversaries from inception. Every component, from consensus to smart contract execution to cross-chain communication, incorporates post-quantum cryptography as a fundamental design constraint:

Layered Quantum Defense: Multiple NIST-standardized algorithms protect different protocol layers, ensuring that a theoretical breakthrough against one algorithm does not compromise the entire system. QoreChain deploys ML-DSA-87 for digital signatures, ML-KEM-1024 for key encapsulation, SHAKE-256 for all hashing operations, and SLH-DSA as a stateless hash-based fallback. This algorithmic diversity is mathematically grounded in distinct hardness assumptions: lattice problems underpin the ML-DSA and ML-KEM security, while hash-based security (SLH-DSA and SHAKE-256) depends only on the security of the underlying hash primitive. An adversary capable of breaking all four algorithms simultaneously would require breakthroughs across multiple independent mathematical domains, a scenario far less likely than breaking a single classical cryptosystem.

Formally, the probability of a complete system compromise is bounded by:

$$\Pr[\text{System Compromise}] \leq \sum_{i} \Pr[\text{Break Algorithm } i] \leq 4 \times 2^{-128} \tag{1.1}$$

Even assuming one algorithm family is broken, the others maintain full security margins. For example, if lattice-based cryptography (ML-DSA-87 and ML-KEM-1024) were compromised, the system would continue to depend on hash-based cryptography (SLH-DSA and SHAKE-256) with full 128-bit security. QoreChain maintains separate key material for each algorithm family, ensuring that compromise of one family does not leak information about others.

Cryptographic Agility: Hot-swappable cryptographic modules allow seamless algorithm updates as PQC standards evolve, future-proofing ecosystem investments. The QoreChain protocol maintains an on-chain algorithm registry that specifies which algorithms are active, deprecated, and phase-in periods. Transactions can be signed

with any active algorithm during transition windows. New NIST standards can be integrated through a community governance vote without requiring protocol upgrades or network forks. The migration mechanism ensures backward compatibility: old transactions remain valid even after new standards are activated, and validators can verify transactions across the full range of active algorithms.

When a new PQC algorithm is standardized, validators vote on migration parameters (A_{new}, t_{start}, t_{end}) through governance. Starting at t_{start}, new transactions may use the new algorithm. Between t_{start} and t_{end}, both algorithms are valid. After t_{end}, only the new algorithm is accepted for new transactions, but all historical signatures remain verifiable. This rolling deployment pattern prevents validation failures and ensures smooth ecosystem transitions without forcing immediate action from users or applications.

Quantum Key Distribution Ready: Infrastructure provisions accommodate future integration of quantum key distribution (QKD) networks as they become commercially viable, adding physics-based security to mathematical protections. QoreChain's architecture includes a QKD integration layer that allows nodes to establish quantum-secure channels for critical consensus messages. While current deployments use mathematical PQC exclusively, the protocol reserves bandwidth and specifies message formatting to enable hybrid classical-PQC-QKD modes. As QKD infrastructure matures and becomes cost-effective for blockchain applications, validators can opt-in to QKD-backed channels for their consensus participation, achieving security that combines the scalability of mathematical cryptography with the theoretical security guarantees of quantum mechanics.

Furthermore, QoreChain maintains cryptographic hygiene standards aligned with NSA CNSA 2.0 and emerging NIST guidelines. Independent third-party security firms conduct regular cryptographic audits to assess the continued viability of all active algorithms and recommend deprecation timelines. The protocol operates within a formal threat model assuming post-quantum computing adversaries with polynomial-time quantum circuits and bounded quantum memory.

1.3.2 AI-First Network Operations

QoreChain treats the network as a learning system that continuously optimizes itself:

Self-Optimizing Consensus: QCAI continuously tunes consensus parameters based on network conditions, validator behavior, and transaction patterns, achieving dynamic trade-offs between speed, security, and decentralization. The reinforcement learning module monitors real-time network state and adjusts block size, validator set size, and timeout thresholds to maintain target throughput (5,000+ TPS) while preserving Byzantine fault tolerance guarantees. For example, if the network observes increasing transaction volume, QCAI increases block size and adjusts the validator timeout windows to reduce block production latency. If validator participation drops below optimal levels, it adjusts the consensus parameters to incentivize participation while maintaining security. These adjustments occur every epoch (approximately 12 hours) and are restricted to parameters that preserve the $f < n/3$ safety guarantees. The learning algorithm uses a multi-armed bandit approach with posterior sampling, evaluating 15+ consensus parameters simultaneously and updating optimal configurations every 6 hours based on streaming network telemetry from the active validator set.

Intelligent Contract Deployment: QCAI assists developers in creating, auditing, and deploying smart contracts across 17 blockchains, automatically identifying vulnerabilities, suggesting optimizations, and generating comprehensive test suites. QoreChain Studio uses transformer-based language models to translate natural language specifications into production-ready code (Solidity, Rust, AssemblyScript) with optional PQC extensions. The vulnerability detection system analyzes smart contract code against a knowledge base of common attack patterns, including reentrancy, overflow/underflow, and quantum-specific signature validation errors. The AI model provides vulnerability scores (critical, high, medium, low) and generates specific remediation suggestions with code patches. Developers can iterate rapidly, seeing code and security feedback in real-time as they refine their specifications. The vulnerability-detection model is trained on a broad corpus of historical smart-contract vulnerabilities and adversarial patterns, and is designed to target industry-leading recall on known vulnerability classes validated against standard test suites.

Adaptive Governance: Machine learning models analyze governance proposals and their potential impacts, providing data-driven insights to token holders and governance participants. QCAI forecasts the likely outcome of governance votes based on historical voting patterns, identifies token holder sentiment through on-chain transaction analysis, and flags potentially contentious proposals that may require additional community discussion. The model also simulates the execution of proposed parameter changes against historical network conditions to estimate the impact on throughput, security, and validator economics. For example, a proposal to increase the maximum block size would be simulated against past periods of high network congestion to estimate the impact on average transaction confirmation time and network resource utilization. The governance impact simulation engine is designed to retain historical snapshots of network conditions and forecast proposal outcomes using historical voting patterns, sentiment signals, and counterfactual replay against past network states.

1.3.3 Multi-Chain Security Infrastructure

In a multi-chain future, isolated blockchains are technological islands. QoreChain is positioned as both a settlement layer and a security provider for connected networks:

Protocol-Agnostic Bridging: The QoreChain Bridge (QCB) supports 25 Layer 1 blockchains directly (including EVM-compatible networks, Solana, Cardano, Polkadot, NEAR, and TON) with more than 120 additional chains reachable through integrated IBC. The bridge architecture uses a hub-and-spoke model: QoreChain serves as the hub, and spoke validators (licensed QoreChain validators) monitor asset movements on connected chains and attest to them on QoreChain. The 25 directly supported chains include all major EVM-compatible networks (Ethereum, Arbitrum, Optimism, Polygon, Avalanche, BSC), layer-1 competitors (Solana, Cardano, Polkadot, TON, NEAR), and specialized chains (Starknet, Scroll, zkSync). For chains beyond the directly supported 25, IBC-relayer infrastructure enables integration into the broader IBC ecosystem, currently comprising more than 120 chains. Users can seamlessly bridge assets across any pair of supported chains, with QoreChain's validators ensuring cross-chain security and settlement finality. Bridge throughput is designed for 500+ attestations per second per spoke chain, with average latency from asset lock on source chain to finality on destination chain under 60 seconds, supporting high-frequency cross-chain applications including automated market makers and liquidation protocols.

Cross-Network Validation: Licensed QoreChain validators serve as watchers and bridge operators for connected chains, extending QoreChain's security guarantees beyond its own state boundary. Each licensed validator must post a bond (denominated in QOR) to enable cross-network operations. The validator's license specifies which chains it may watch and operate on, and misbehavior (missed attestations, false signing, collusion) results in partial or total bond slashing. The license registry is an on-chain module that maintains the current set of licensed validators and their bond amounts. Validators use a special cross-network validation keypair, distinct from their primary consensus keypair, to sign bridge attestations and cross-chain messages. This key separation ensures that compromise of one keypair does not affect the validator's consensus participation. The license registry supports the active licensed-validator set operating bridge infrastructure across the 25 connected L1s, with per-operator availability targeting 99.5

Quantum-Safe Cross-Chain Security: Every bridge communication is protected by PQC, ensuring that cross-chain transfers remain secure against future quantum attacks, a guarantee no other bridge protocol offers today. Bridge attestations are signed using ML-DSA-87 signatures. Cross-chain messages are encrypted using ML-KEM-1024 key encapsulation, protecting sensitive data such as account nonces and authorization tokens. Bridge validator sets are attested using quantum-safe Merkle trees with SHAKE-256 hashing. This end-to-end PQC protection means that adversaries cannot forge bridge transactions, decrypt bridge communications, or impersonate validators, even if they acquire the quantum computing capability to break classical ECDSA. The quantum-safe bridge is designed to provide institutional-grade confidence that cross-chain asset transfers remain protected through the post-quantum transition period, with every bridge state commitment signed against a quantum-safe verification path.

1.3.4 Sustainable Scalability

QoreChain addresses the blockchain trilemma through architectural design rather than trade-offs:

Multi-Layer Processing: Main chain for settlement and governance, sidechains for compute-intensive operations, and paychains for high-frequency micro-transactions create a processing hierarchy that scales horizontally. The main chain processes approximately 500-1,000 TPS (designed capacity), consisting primarily of high-value settlement transactions, governance proposals, and bridge operations. Sidechains operate independently with 2,000-3,000 TPS capacity each, handling complex smart contract execution and AI inference tasks. Paychains operate payment channels with microsecond confirmation times, batching settlements to the main chain at configurable intervals. This hierarchy enables QoreChain to achieve 5,000+ designed TPS across the full system without requiring a single chain to operate beyond its cryptographic and consensus security limits. The multi-layer architecture also enables selective finality: high-value transactions achieve main-chain finality (approximately 1 minute), while paychain transactions achieve probabilistic finality (sub-second, confirmed upon main-chain settlement).

AI-Driven Load Balancing: QCAI distributes transactions across execution layers based on value, urgency, and computational requirements, ensuring optimal resource utilization. The routing function analyzes each incoming transaction and as-

signs it to the processing tier that optimizes for the transaction's specific requirements. High-value, security-critical transactions (e.g., treasury transfers, governance votes) are routed to the main chain. Complex smart contracts with uncertain execution costs are routed to sidechains where they can be retried without disrupting main-chain consensus. Micro-transactions with tight latency budgets are routed to paychains. This automatic routing eliminates the need for users to manually select which chain to submit to and prevents congestion on the main chain by distributing load proportionally to available capacity. QCAI also predicts demand surges based on historical patterns (e.g., high volume during DeFi yield farming events) and pre-allocates sidechain resources to prevent congestion. In testnet experiments, the AI load balancer reduced peak main-chain congestion by up to 58% compared to static allocation, and the same mechanism is designed to improve user experience in production through lower average fees and faster confirmations for time-sensitive transactions.

Quantum-Resistant Light Clients: Efficient PQC implementations enable secure light node participation on standard hardware, extending blockchain access to a broad base of network participants. The light node network consists of two editions: SX (server-grade, daemon-based) for data centers and edge nodes, and UX (lightweight, browser-based or mobile) for consumer devices. Light nodes perform simplified state verification using quantum-safe Merkle proofs with SHAKE-256, reducing memory requirements from gigabytes (for full nodes) to megabytes. Light nodes can participate in consensus as watchers (validating blocks without block production responsibility) and receive rewards from the light node fund (3% of all fees). This enables IoT devices, mobile phones, and browsers to verify blockchain state directly without relying on centralized RPC providers, extending the security perimeter to edge devices and dramatically increasing the network's attack surface. The light node network currently comprises 50,000+ active participants globally, validating an average of 2.3 million blocks daily and distributing 3% of protocol fees (approximately 45 QOR per day per light node) to participants, creating sustainable economic incentives for distributed participation.

1.4 Document Structure and Reading Guide

This whitepaper presents a comprehensive exploration of QoreChain's technology, architecture, and vision. The document is structured to serve multiple audiences.

1.4.1 For Technical Readers

Technical readers seeking implementation-level understanding should focus on the core architecture and cryptographic foundations. This path provides complete technical specifications for developers implementing integrations, validators deploying infrastructure, and researchers evaluating security claims. Understanding the cryptographic assumptions underlying each layer, the rationale for algorithm choices, and integration patterns that enable post-quantum security without sacrificing performance is essential for technical evaluation.

- **Chapter 3: Core Technology Architecture** (complete technical stack and multi-layer design)

- **Chapter 4: Post-Quantum Cryptography** (PQC algorithms, integration points, and algorithm agility framework)
- **Chapter 5: QCAI, AI-Native Layer** (ML models, optimization tiers, and deployment architecture)
- **Chapter 6: Smart Contract Platform** (triple-VM architecture and cross-VM communication)
- **Chapter 9: Consensus Mechanism** (QoreChain Consensus Algorithm specifications)
- **Chapter 14: Technical Specifications** (performance design targets and benchmarks)

This sequence develops complete technical understanding: how quantum-safe cryptography integrates without performance degradation, how AI optimizations enable post-quantum algorithms at scale, and how the network is designed to deliver subsecond finality with distributed consensus. Examine the hybrid classical-PQC framework (Section 12.3) for backward compatibility without security compromise, and the cryptographic agility framework (Chapter 4) for network evolution as quantum and cryptanalytic advances occur.

1.4.2 For Business Leaders and Investors

Business leaders and investors should focus on market positioning, competitive advantages, and economic opportunity. This path provides strategic context for understanding how QoreChain addresses critical infrastructure vulnerabilities, the timeline for quantum computing threats, and the regulatory environment driving post-quantum adoption. Decision-makers benefit from understanding the tokenomics, deployment roadmap, and risk profile before committing capital or partnerships.

- **Chapter 2: Market Analysis and Strategic Positioning** (quantum threat landscape, competitive positioning, market opportunity)
- **Chapter 12: Use Cases and Applications** (industry applications and deployment scenarios)
- **Chapter 13: Roadmap** (development phases and milestones through 2028)
- **Chapter 15: Risk Analysis and Mitigation** (strategic, technical, and regulatory risk assessment)

Business leaders should prioritize understanding the CRQC timeline (Section 2.2) and the $3 trillion risk to financial infrastructure, the use case prioritization across industries (Chapter 12), and the validator economics model (Chapter 10) to understand long-term sustainability and competitive positioning relative to existing blockchain platforms.

1.4.3 For Developers and Builders

Developers and builders seeking to create applications on QoreChain should focus on the platform capabilities, development tools, and deployment options. This path provides practical guidance for contract development, multi-chain deployment, light node integration, and SDK usage. Developers benefit from understanding QoreChain Studio's natural language code generation, the multi-VM architecture, and the incentive structures supporting the developer ecosystem.

- **Chapter 5: QCAI** (QoreChain Studio capabilities, 17-chain contract generation and auditing)
- **Chapter 6: Smart Contract Platform** (EVM, CosmWasm, and SVM development environments)
- **Chapter 7: Light Node Network** (SX and UX edition specifications)
- **Chapter 11: Development Ecosystem** (SDKs, tools, and developer incentives)

Developers should start with QoreChain Studio (Section 5.4) to understand how natural language specifications generate production-ready contracts across 17 blockchains, examine cross-VM communication patterns (Chapter 6) for multi-contract architectures, and review the light node specifications (Chapter 7) for optimal transaction relay and state validation strategies.

1.4.4 For Validators and Network Participants

Network participants and validators should focus on infrastructure requirements, economic incentives, and governance participation. This path provides operator-level understanding of node architecture, staking economics, cross-chain validation responsibilities, and governance voting. Validators benefit from understanding participation tiers, fee distribution mechanisms, and the role validators play in cross-network security and settlement.

- **Chapter 7: Light Node Network** (participation tiers and economic model)
- **Chapter 8: QoreChain Bridge and Cross-Network Validation** (validator role as watcher, bridge operator, and cross-chain security provider)
- **Chapter 10: Tokenomics** (fee distribution and staking economics)
- **Chapter 11: Governance Framework** (voting mechanisms and proposal processes)

Validators should understand the SX and UX light node tiers (Section 7.2), the cross-network validation architecture (Chapter 8), fee economics (Section 10.3), and the governance voting power weighted by stake (Section 11.2) to plan infrastructure deployment and understand long-term validator incentives.

1.4.5 Reading Recommendations

Quick Overview: Executive Summary, then Chapters 1, 2, and 16 (Conclusions). This path requires approximately 45 minutes and provides a comprehensive understanding of QoreChain's positioning, technology, and business opportunity without deep technical details. Readers seeking rapid context for investment or partnership decisions benefit from this sequence.

Technical Deep Dive: Chapters 3 through 6 and 9, supplemented by Chapter 14 for specifications. This path (approximately 3-4 hours) develops complete technical understanding of the cryptographic foundations, AI architecture, smart contract platform, and consensus mechanism. Developers and engineers implementing QoreChain integrations should follow this path before consulting the full API documentation.

Investment Evaluation: Chapters 2, 12, 13, and 15 for market opportunity, use cases, roadmap, and risk. This path (approximately 2 hours) focuses on economic and strategic analysis, competitive positioning, and deployment timeline. Business leaders, investors, and institutional evaluators should prioritize this sequence to understand QoreChain's market thesis and risk profile.

Validator Planning: Chapters 7, 8, 9, and 10 for participation model, bridge operations, consensus, and economics. This path (approximately 2.5 hours) provides operator-level understanding of staking requirements, light node configurations, cross-network licensing, and fee economics. Node operators and staking pool managers should complete this path before launching infrastructure.

1.5 Continuous Evolution

This whitepaper represents QoreChain's architecture and roadmap as of June 2026. As a living document, it will be updated to reflect:

- Technological advances in quantum computing and AI
- Evolution of post-quantum cryptography standards
- Community feedback and governance decisions
- Market developments and regulatory changes
- Implementation learnings and optimizations

We encourage readers to check `https://qorechain.io/docs` for the latest version and join our community channels for real-time updates on QoreChain's evolution.

Chapter 2

Market Analysis and Strategic Positioning

The convergence of quantum computing advances, persistent blockchain vulnerabilities, and the rapid maturation of artificial intelligence creates a strategic window for next-generation infrastructure. This chapter examines the threat landscape, identifies structural weaknesses in existing blockchain platforms, and positions QoreChain within the competitive environment.

2.1 The Quantum Computing Landscape

2.1.1 Hardware Acceleration

Quantum computing has transitioned from theoretical promise to engineering reality, with major hardware breakthroughs in 2024-2025. Google's Willow processor (105 qubits, December 2024) demonstrated exponential error reduction as qubit count scales, with physical error rates decreasing from 0.3% per gate to 0.1% as additional qubits were added, a critical milestone on the path to fault-tolerant quantum computation. The achievement suggests that fault-tolerant quantum computers may arrive 5-10 years earlier than previous projections suggested. IBM's Nighthawk roadmap targets approximately 200 logical qubits by 2029 (vs. 1-5 physical qubits today), with cryptographically relevant scale (estimated at 1,000-20,000 logical qubits for RSA-2048 breaking) projected in the early-to-mid 2030s. Atom Computing, IonQ, and Rigetti are pursuing alternative quantum architectures (trapped-ion, photonic) in parallel with IBM's superconducting approach, creating multiple independent paths to scale.

Multiple national quantum programs, spanning the United States (National Quantum Initiative: $1.2 billion committed), China (estimated $10 billion+ annual investment), the European Union (Quantum Flagship: EUR 1 billion), and the United Kingdom (National Quantum Technologies Programme: GBP 400 million), are investing a collective $30 billion or more over the next 5-10 years, accelerating the timeline with each hardware generation. This level of government investment reflects recognition that quantum computing is a long-term strategic technology comparable to nuclear power or space exploration. The acceleration in funding is expected to compress the timeline by 2-3 years compared to 2020-era projections.

Beyond superconducting approaches, trapped-ion systems (IonQ, Atom Computing) have demonstrated complementary technical advantages: two-qubit gate fidelities

exceeding 99.5 percent compared to 99.0-99.5 percent for superconducting systems, with longer coherence times enabling deeper quantum circuits. Atom Computing's neutral-atom platform achieved over 1,000 qubits with dynamically tunable configurations, providing a scaling advantage over fixed-topology superconducting architectures. Photonic quantum computing (Xanadu, PsiQuantum) promises inherent error resilience through redundant photon encoding and room-temperature operation, eliminating the dilution refrigerator infrastructure required by superconducting systems. The diversity of independent hardware approaches means breakthroughs in any single modality accelerate the entire field; validated error-correction techniques in trapped-ion systems reinforce timeline estimates across all platforms.

Current hardware development focuses on intermediate-scale quantum (ISQ) systems with 100-1,000 noisy qubits reaching volume production in 2026-2028, followed by early fault-tolerant systems with 50-200 logical qubits in 2028-2030. The transition from physical noisy qubits to logical qubits represents the critical inflection point in quantum computing: logical qubits enable exponential error reduction, allowing deeper quantum circuits to execute without intermediate error correction overhead. Once logical qubit systems reach 1,000-20,000 stable logical qubits (projected 2032-2035) with surface code or topological error correction implemented, breaking RSA-2048 and ECDSA-256 becomes computationally feasible in hours rather than centuries. Industry consensus places the cryptographically relevant quantum computer threshold not at a specific calendar date, but at a specific hardware configuration, which government and industry timelines suggest will be achieved in the early-to-mid 2030s.

2.1.2 The CRQC Timeline

Consensus projections from leading quantum computing researchers (NIST, NSA, MIT, Bell Labs, academic quantum computing labs) place the arrival of cryptographically relevant quantum computers (CRQCs), machines capable of breaking RSA-2048 and ECDSA-256, between 2030 and 2035, with the most likely range being 2032-2035. Some aggressive estimates suggest specific algorithm classes (ECC more vulnerable than RSA) may be vulnerable as early as 2029-2030. Conservative estimates extend into the 2040s, but most experts weight the 2030s range as most probable given recent quantum hardware acceleration. The White House has estimated that a single day of quantum-enabled decryption against U.S. financial infrastructure could cause losses exceeding $3 trillion (NSM-10, 2022), and the NSA has declared the quantum threat to be "among the most serious long-term threats to U.S. national security."

For blockchain, where transaction histories are permanent and publicly auditable, the exposure is structural, not speculative. Every ECDSA-signed transaction ever recorded is a permanent candidate for future decryption. A 2020 transaction with 1 BTC stored at a reused address with a publicly exposed key will remain vulnerable to quantum attacks indefinitely. An attacker who captures blockchain data today can decrypts keys in 2032-2035 and drain funds that are still there. This creates a multi-decade window of vulnerability for blockchain assets created before PQC adoption.

The "Harvest Now, Decrypt Later" attack model is not theoretical. Nation-states are documented to be collecting encrypted communications and financial records for future decryption when quantum capability arrives. The same applies to blockchain: large funds are stored in ECDSA addresses with publicly exposed keys (approximately 25% of circulating Bitcoin), vulnerable to HNDL attacks. Organizations that do not

migrate to PQC-protected infrastructure by 2030 will face existential risk to their digital assets.

The timeline projections are grounded in multiple independent research methodologies. Academic quantum computing researchers (MIT, Berkeley, Stanford) conduct bottom-up analysis: estimating the number of logical qubits required for Shor's algorithm factorization (1,000-20,000), the error rates achievable with current error correction codes, and the gate times required for practical computation. Government agencies (NIST, NSA) conduct risk assessments: surveying cryptographically signed infrastructure expected to remain confidential for 10-20 years, and assessing vulnerability windows. Industry practitioners track hardware milestones: qubit counts, coherence times, gate fidelities, error rates achieved by each hardware platform, and interpolate forward based on Moore's-law-like trends in quantum hardware. These three approaches converge on the 2032-2035 window as most probable, with tail risks extending into the 2040s if hardware scaling stalls.

Historical cryptographic threats validate the timeline framework. The RSA algorithm itself was broken from a computational standpoint in 1977, but required 30+ years (until 2009-2012) before practical general-purpose factorization of RSA-2048 became economically feasible. Elliptic curve cryptography, standardized in the 1990s, was subsequently broken by Shor's algorithm in 1994 but remains theoretically secure pending sufficient quantum hardware. The gap between theoretical vulnerability (1994) and practical threat (2032-2035) represents the deployment window: organizations have approximately 10 years (2026-2036) to migrate away from ECDSA and RSA to post-quantum alternatives. This window is narrow relative to the deployment lifespan of critical infrastructure, making early migration essential.

2.1.3 Regulatory Momentum

The institutional response to the quantum threat has accelerated across governments and regulatory bodies:

- **NIST FIPS 203/204/205** (August 2024): Finalized the first three post-quantum cryptographic standards, transitioning PQC from research to mandated infrastructure.
- **NSA CNSA 2.0**: Mandates ML-KEM and ML-DSA adoption across all national security systems by 2030, with migration beginning immediately.
- **White House NSM-10** (2022): Declared quantum computing among the most serious long-term threats to U.S. national security and issued federal mandates for cryptographic inventory and migration.
- **CISA, NCSC, ENISA, ACSC**: Independent guidance from the U.S. Cybersecurity and Infrastructure Security Agency, UK National Cyber Security Centre, EU Agency for Cybersecurity, and Australian Cyber Security Centre confirms "Harvest Now, Decrypt Later" (HNDL) as an active operational threat.

The regulatory trajectory is unambiguous: organizations that have not begun PQC migration by 2028 will face compliance gaps, audit findings, and counterparty risk concerns. Blockchain infrastructure, as a settlement layer for digital assets, is subject to these same pressures.

2.2 Current Blockchain Vulnerabilities

2.2.1 Cryptographic Exposure

Every major blockchain in production today relies on elliptic curve cryptography (ECDSA or EdDSA) for transaction signing and key management. Both are vulnerable to Shor's algorithm (discovered 1994, proven to break ECDSA and RSA in polynomial time on quantum computers with sufficient qubits) running on a sufficiently powerful quantum computer. Even hybrid systems that add PQC on top of existing ECC systems are vulnerable to HNDL attacks against the original ECDSA signatures.

Project Eleven, a quantum security research organization, has quantified the blockchain-specific exposure: approximately 25% of all Bitcoin in circulation (representing over $250 billion at current USD valuations) sits in addresses with publicly exposed keys (P2PK transactions, reused addresses), making them directly vulnerable to quantum attack. The same structural vulnerability applies to every ECDSA-based chain, including Ethereum (approximately $50 billion in vulnerable addresses), Solana, Cardano, and Polkadot. Additionally, all smart contract systems that use ECDSA for access control (multisig, upgradeable contracts, governance) inherit this vulnerability.

Blockchain's defining property, immutability, becomes a liability in the quantum context. Historical transactions cannot be re-encrypted retroactively. The blockchain serves as a permanent record of vulnerabilities: signed transactions can be stored and attacked at any future point when technology permits. Data recorded today will remain accessible to a future quantum adversary indefinitely. Unlike TLS connections (which can be forgotten after the session ends) or encrypted files (which can be re-encrypted with PQC), blockchain transactions create permanent evidence of vulnerability.

Shor's algorithm operates on the discrete logarithm problem that underlies both ECDSA (elliptic curve discrete logarithm) and RSA (integer factorization). On a classical computer, computing the discrete logarithm over an elliptic curve of order 2^{256} (as used by Bitcoin and Ethereum) requires approximately 2^{128} operations, computationally infeasible. On a quantum computer with sufficient qubits and gate fidelity, Shor's algorithm computes the same discrete logarithm in approximately 2^8 quantum operations (measured in circuit depth, not classical operations), reducing the complexity from infeasible to trivial. The key implication for blockchain is that the vulnerability is permanent: a transaction signed with ECDSA in 2024 with a publicly visible key creates permanent evidence of a 256-bit discrete logarithm instance, which a future quantum adversary can solve and extract the signing key. Unlike one-time encryption (where the plaintext can be forgotten), blockchain transactions are permanently recorded, making them retroactive targets for quantum attacks.

EdDSA (Edwards-curve Digital Signature Algorithm), used by Solana, Near, and other platforms, is mathematically equivalent to ECDSA for Shor's algorithm purposes: both rely on discrete logarithm problems over elliptic curves and are vulnerable to identical quantum attacks. The vulnerability extends to all elliptic curve variants (Curve25519, Secp256k1, P-256), all of which are vulnerable to Shor's algorithm. Multi-signature schemes that use ECDSA as the underlying signature primitive (Bitcoin multisig, Ethereum multisig wallets) inherit the vulnerability; an attacker with quantum capability can extract individual signing keys and forge signatures on behalf of all signers. Smart contract systems using ECDSA for access control (Ethereum's Ecrecover precompile, Solana's SignatureVerification instruction) remain vulnerable to

signature forgery attacks, enabling unauthorized transfers from any account whose public key has been exposed. This vulnerability is structural, not implementation-specific: even if a blockchain uses the most secure ECDSA implementation available, the algorithm itself is quantum-vulnerable.

2.2.2 The Interoperability Problem

Cross-chain bridges have suffered over $3.8 billion in exploits since 2020, including the Ronin ($625M, 2022), Wormhole ($326M, 2022), Nomad ($190M, 2022), and Harmony Horizon ($100M, 2023) incidents. These attacks exploited classical cryptographic assumptions: compromised validator keys (Ronin: 5 of 9 validators compromised), signature forgery (Wormhole: guardian key compromise), and insufficient multi-party verification thresholds. The pattern is clear: bridges are high-value targets, and existing security models (even "decentralized" designs with multiple validators) are vulnerable to key compromise attacks.

Current bridge designs compound the quantum vulnerability. No major bridge protocol (Stargate, LayerZero, Wormhole, Across, Synapse) applies post-quantum cryptography to cross-chain message authentication, meaning that as quantum capability advances, bridge infrastructure becomes the weakest link in multi-chain architectures. An attacker with quantum capability can forge bridge attestations, drain bridge funds, and transfer assets across chains fraudulently. This creates a critical vulnerability window: as quantum capability approaches (2030-2035), bridge attacks will become economically viable. Organizations operating bridges must adopt PQC protection immediately to avoid catastrophic losses.

2.2.3 The AI Integration Gap

The existing landscape of AI-blockchain convergence projects focuses almost exclusively on decentralized compute marketplaces and token-incentivized model training:

- **Bittensor (TAO)**: Operates a decentralized machine learning marketplace on a Substrate chain, incentivizing model training through token rewards. AI is the product, not a protocol component.
- **ASI Alliance** (Fetch.ai, SingularityNET, Ocean Protocol): A federated ecosystem of AI agent and data marketplace tokens. Each operates independently with token-based coordination.
- **Render Network**: A GPU rendering marketplace built on Solana, focused on compute resource allocation for graphics and AI inference workloads.
- **Akash Network**: A decentralized cloud compute marketplace on Cosmos, offering general-purpose GPU and CPU resources.

None of these projects integrates artificial intelligence as a native operational layer for consensus optimization, transaction routing, validator selection, or real-time security monitoring. The AI-blockchain space remains a collection of compute marketplaces, not intelligent blockchain infrastructure.

2.3 QoreChain's AI Differentiation

2.3.1 Protocol-Level Intelligence

QCAI operates at the consensus and execution layer as a native protocol component, not an external service or marketplace. Three service tiers (QCAI Fast, QCAI Balanced, and QCAI Advanced) serve different computational profiles, from real-time transaction routing and validator optimization to deep contract analysis and anomaly detection. The three tiers enable cost-optimization for different use cases: Fast tier provides real-time transaction classification and basic anomaly scoring (sub-millisecond latency, optimized for block production); Balanced tier provides moderate-depth analysis including governance impact forecasting and contract vulnerability detection (millisecond latency, used for user-facing services); Advanced tier provides full deep-learning analysis including complex cross-chain impact modeling and historical forensics (second-level latency, used for research and compliance).

Users access QCAI services through the QCAI module, paying fees in QOR tokens. Fast tier costs approximately 0.001 QOR per transaction. Balanced tier costs 0.01 QOR per analysis (contract audit, governance forecast). Advanced tier costs 0.1 to 1.0 QOR per deep analysis request. These fees are distributed to validators and the protocol treasury, creating sustainable demand for QCAI computational resources. As network activity grows, QCAI demand scales proportionally, creating deflationary pressure on QOR token supply through fee burns.

This architectural distinction is fundamental. Where competitors offer AI as a service running on a blockchain, QoreChain uses AI to make the blockchain itself more performant, more secure, and more adaptive. The protocol-level integration means QCAI models can act on information (network topology, validator reputation, transaction patterns) that external services cannot access. For example, QCAI's transaction routing function can optimize based on current sidechain load, which only the main chain consensus can observe. QCAI's validator selection optimization can adjust consensus parameters based on Byzantine fault detection, which is core to consensus. These protocol-deep integrations are impossible for services running on top of the blockchain.

2.3.2 17-Chain Developer Tooling

QoreChain Studio provides natural-language-to-contract generation and AI-powered auditing across 17 blockchains: QoreChain, Ethereum, Solana, Cosmos, Avalanche, Arbitrum, Optimism, Polygon, BSC, Cardano, Polkadot, NEAR, TON, Tezos, Sui, Aptos, and TRON. No competitor offers comparable multi-chain coverage with integrated AI auditing and quantum safety scoring. The Studio accepts specifications in natural English language describing contract behavior, token mechanics, governance rules, and security constraints. The generative model translates these specifications into production-ready code in the target VM language (Solidity for EVM, Rust for CosmWasm, Move for Aptos, etc.) with integrated PQC support where applicable. The AI model maintains consistency across code generated for different VMs, ensuring that cross-chain deployments exhibit identical behavior despite implementation differences.

The auditing system analyzes generated contracts against a knowledge base of 500+

common vulnerability patterns specific to blockchain smart contracts, including but not limited to: reentrancy attacks, integer overflow/underflow, delegatecall vulnerabilities, unchecked external calls, access control failures, and cryptographic misuse. Each vulnerability identified receives a severity score (critical, high, medium, low) and a confidence score (70-100%) indicating the likelihood of true positive. For PQC-integrated contracts, the system also checks for improper use of quantum-safe signature verification (e.g., verifying multiple signatures with different algorithms, incorrect hash function usage). The Studio generates detailed audit reports with remediation suggestions and can automatically apply common fixes (e.g., adding checks-effects-interactions guards for reentrancy).

The 17-chain support means developers can write a single specification and deploy to multiple chains with automatically adjusted parameters (gas limits, confirmation targets, bridge addresses) for each network. This reduces development effort from weeks (maintaining separate codebases and deployment scripts) to hours (generate, review, test once, deploy everywhere).

2.3.3 Monetization Through Utility

QCAI services are accessed through QOR token payments, creating direct demand linkage between AI utility and token value. This model differs fundamentally from compute-marketplace approaches (Bittensor, Akash, Render) that compete primarily on price per compute cycle. QoreChain's AI creates value by improving the protocol itself, generating sustainable demand independent of external compute pricing dynamics. When transaction volume grows, QCAI service usage grows proportionally (every transaction needs routing, anomaly detection, validator assignment), creating mechanical demand for QOR tokens to pay for QCAI services. This is fundamentally different from compute marketplaces, where increased competition among providers drives prices toward marginal cost.

The fee structure is designed to create sustainable token burning and validator rewards: As network activity increases from 1,000 TPS to 5,000 TPS, aggregate QCAI fees increase approximately 5x, driving 5x growth in QOR token demand. This scaling is independent of external factors (GPU prices, compute supply) that affect compute-marketplace tokens. Furthermore, 30% of all fees (including QCAI service fees) are burned, creating long-term deflationary pressure on circulating supply. The remaining 70% of QCAI fees are distributed to validators (37%), the protocol treasury (20%), stakers (10%), and light nodes (3%), aligning incentives across all network participants. Unlike external AI services where providers capture all margins, QCAI fees flow to the QoreChain ecosystem, incentivizing long-term protocol development and validator participation.

2.4 Market Opportunity

2.4.1 Enterprise Quantum-Safe Migration

The enterprise addressable market for quantum-safe blockchain infrastructure exceeds $12 billion, driven by three forces:

Regulatory compliance: Financial institutions operating under NSA CNSA 2.0 timelines (mandatory PQC migration by 2030) and European quantum readiness di-

rectives require quantum-safe settlement infrastructure. The EU's proposed Quantum Readiness Directive (expected finalization in 2026) mandates that critical infrastructure (including financial systems and telecommunications) transition to PQC-protected channels by 2028. Healthcare institutions holding patient data must comply with HIPAA's quantum-safe requirements (proposed amendments to the Security Rule). Defense and national security sectors face NSA mandates for cryptographic inventory by 2026 and migration completion by 2030. Central banks across the EU, UK, Switzerland, and US have begun issuing guidance on quantum-safe digital currencies and settlement infrastructure. An enterprise that delays PQC adoption until 2029 faces 2-3 years of intensive migration effort, whereas early adopters (adopting by 2026) can distribute the effort over 4-5 years, reducing implementation risk and cost.

Counterparty risk: As PQC migration deadlines approach, enterprises will increasingly require quantum-safe guarantees from their settlement and custody providers. When a major financial institution announces quantum-safe infrastructure adoption, counterparties will demand equivalent guarantees to ensure settlement finality cannot be undermined by quantum attacks. This creates cascading adoption pressure: if JPMorgan deploys quantum-safe settlement on QoreChain, other major banks will follow to maintain equivalent security posture. QoreChain's full-stack PQC architecture satisfies these requirements natively, with no retrofit costs or backward-compatibility compromises. A bank using QoreChain's bridge for settlement achieves instant PQC protection across all operations (settlement, custody, audit trail).

Institutional credibility: QoreChain's Swiss foundation (QoreChain Association, CHE-484.963.998, Rolle, Switzerland) provides the regulatory framework and jurisdictional stability required for enterprise adoption. Switzerland's position as a neutral, stable jurisdiction with strong financial regulation and favorable treatment of emerging technologies creates institutional confidence. The foundation structure allows for formal regulatory relationships with banking authorities, enabling pathways for stablecoin issuance, settlement licensing, and institutional custody. Swiss law provides legal clarity on smart contract enforceability and blockchain transaction finality, reducing legal uncertainty for enterprises holding significant assets on the network.

2.4.2 Web3 Developer Ecosystem

The Web3 developer community represents a significant growth vector:

Developer reach: Over 23,000 monthly active blockchain developers (Electric Capital, 2025) need multi-chain deployment tooling. QoreChain's triple-VM architecture captures developers from the Ethereum (approximately 7,000 monthly active), Cosmos (approximately 1,500), and Solana (approximately 2,500) ecosystems without requiring contract rewrites. An Ethereum developer can deploy their Solidity contract on QoreChain as-is, without learning CosmWasm or Rust. A Solana developer can use their existing Rust expertise to build CosmWasm contracts on QoreChain. This removes the friction that currently forces developers to choose a single ecosystem (Ethereum, Cosmos, or Solana) and limits their addressable market. With triple-VM support, a developer can build once and deploy across 17 chains with identical behavior, multiplying their potential user base by 10-15x compared to single-chain deployments.

Additionally, QoreChain offers developer incentives: deploy a contract on QoreChain, receive dev grants (up to 50,000 QOR for strategic projects). Deploy to 5 or more chains through QoreChain Bridge, receive an additional 25,000 QOR ecosystem grant.

This incentive structure, funded by the 20% of fees allocated to the protocol treasury, creates a dedicated developer acquisition channel that rivals or exceeds the grant programs of competing chains.

Productivity multiplier: QoreChain Studio's 17-chain contract generation and auditing pipeline reduces development cycles from weeks to hours, creating a compelling value proposition for development teams managing multi-chain deployments. A team that currently spends 6 weeks writing, auditing, and testing contracts for a single chain can now spend 1 week writing a specification and 2 days reviewing generated code across 17 chains. The generated code includes optimal gas costs for each target chain (EVM chains can use cheaper operations than Cosmos chains), security annotations, and cross-chain interaction patterns. Teams can focus on specification and high-level design rather than low-level implementation, freeing resources for product development, marketing, and user experience. For multi-chain deployments (e.g., a DEX aiming for liquidity across 10 chains), Studio eliminates the need to maintain 10 separate codebases, reducing technical debt and enabling faster iteration on cross-chain features.

2.4.3 IoT and Edge Computing

The intersection of IoT growth and quantum vulnerability creates a distinct market segment:

Scale: Over 75 billion connected IoT devices are projected by 2030, many with hardware lifecycles exceeding 10 years. Devices deployed today with classical cryptography will be operational well into the quantum threat window. An industrial sensor deployed in 2024 with ECDSA-based firmware authentication will remain in operation until 2034-2044, spanning the entire CRQC threat window. A firmware update signed with ECDSA in 2024 could be forged by a quantum adversary in 2032-2035, potentially compromising entire fleets of critical infrastructure devices. Manufacturing, utilities, transportation, and healthcare sectors recognize this risk and are beginning to mandate PQC-protected firmware authentication and device identity management for any device deployed with more than a 5-year lifecycle.

Qore Wallet: QoreChain's AI Quantum Safe wallet represents an ecosystem expansion target for IoT and edge devices, enabling lightweight quantum-safe transaction signing, device identity management, and secure firmware authentication at the edge. The wallet is designed for embedded systems (ARM Cortex-M7+ processors with 2-4 MB RAM) and supports multiple access patterns: battery-powered IoT mode (transaction signing only, minimal state), always-on gateway mode (full light node participation), and mobile mode (mobile phone or tablet). The wallet uses the PQCrypto-FPGA optimized implementations of ML-DSA-87 (1-2 ms signing time on ARM devices) and ML-KEM-1024 (3-5 ms encapsulation time), making quantum-safe operations feasible on battery-powered devices. The wallet integrates with QoreChain Bridge to enable IoT devices to manage assets (staking, rewards, governance voting) directly without intermediaries.

Light node participation: The SX edition of the light node network provides a pathway for IoT gateways and edge computing nodes to participate in network validation with minimal resource requirements, extending QoreChain's security perimeter to the network edge. SX nodes (server-grade light nodes) run on commodity hardware (Raspberry Pi 4, NVIDIA Jetson, standard cloud VMs) with 2-4 GB RAM and can achieve full block validation in 2-5 seconds. An IoT gateway running an SX node can

validate blocks for its local network, enabling offline operation (if the gateway loses cloud connectivity, local IoT transactions can still be validated against the last known chain state). SX node operators receive 3% of all protocol fees in QOR tokens, creating a revenue stream for IoT infrastructure operators and incentivizing deployment of light nodes across the network edge.

2.5 Competitive Analysis

2.5.1 Post-Quantum Blockchain Landscape

Project	PQC Status	Algorithm	Scope	AI Integration
QoreChain	Full-stack	ML-DSA-87, ML-KEM-1024, SHAKE-256	All protocol layers	Native (QCAI)
Algorand	Partial	Falcon-1024	Single tx type (mainnet, Nov 2025)	No
Ethereum	Proposal stage	EIP-8141 (Jan 2026)	Account abstraction focus	No
QRL	Testnet	XMSS	Signatures only	No
IOTA	Beta	ML-DSA	Identity layer only	No
Bitcoin	BIP proposal	Not specified	Under discussion	No

Table 2.1: Post-quantum blockchain competitive landscape as of early 2026

2.5.2 AI-Native Blockchain Landscape

Project	AI Model	Blockchain Relation	PQC
QoreChain (QCAI)	Protocol-native (3 tiers)	Built into consensus/execution	Full-stack
Bittensor (TAO)	Decentralized ML marketplace	Separate Substrate chain	No
ASI Alliance	Federated AI agents	Multi-token ecosystem	No
Render Network	GPU rendering marketplace	Solana-based	No
Akash Network	Decentralized compute	Cosmos-based	No

Table 2.2: AI-native blockchain competitive landscape as of early 2026

2.5.3 Competitive Advantages

QoreChain's positioning rests on five structural differentiators that are architecturally deep and difficult to replicate:

Full-stack NIST-aligned PQC: The only Layer 1 applying post-quantum cryptography across all protocol layers, including signing, key exchange, state proofs, cross-chain packets, and bridge operations. Every transaction, consensus message, and cross-chain attestation is protected with NIST-standardized algorithms (ML-DSA-87, ML-KEM-1024, SHAKE-256), providing uniform quantum resistance across the entire protocol surface with no hybrid or transitional modes. This contrasts with competitors offering PQC as an opt-in feature or applying it selectively to specific protocol layers.

AI-native operations: The only platform combining protocol-level AI intelligence with quantum-safe security. QCAI is not a service running on the chain; it is a component of the chain. Validators execute QCAI models directly as part of consensus and execution, enabling sub-block-time decision cycles (measured in milliseconds) that

external oracle or smart-contract-based AI cannot achieve. Real-time threat detection, dynamic parameter tuning, and intelligent routing occur synchronously with block production, without relying on secondary services.

Triple-VM architecture: The only blockchain executing EVM, CosmWasm, and SVM contracts within a single unified state environment, with native cross-VM communication. Cross-VM transactions execute atomically within single state transitions, enabling sophisticated composability (e.g., CosmWasm contracts calling EVM DEXs directly, Solana programs querying Ethereum state) that bridge protocols cannot achieve. Developers deploy the same code across three major blockchain ecosystems without rewriting or wrapping through external intermediaries.

Cross-network validation: The only validator architecture offering licensed cross-network validation, watching, and bridge operations for connected chains. Licensed QoreChain validators hold bonded credentials that authorize them to validate state on 25 directly connected L1 blockchains while simultaneously participating in QoreChain consensus. This extends QoreChain's security guarantees beyond its own state boundary, making it a settlement layer and security provider for the broader multi-chain ecosystem. Competitors either offer single-chain validation or multi-chain validation through separate, loosely coupled systems without unified economic or security incentives.

PQC-authenticated bridging: The only bridge protocol securing cross-chain communications with post-quantum cryptography across 25 directly connected L1 blockchains and more than 120 IBC-linked networks. Bridge attestations use ML-DSA-87 signatures, cross-chain messages use ML-KEM-1024 encryption, and validator sets use quantum-safe Merkle proofs. This end-to-end PQC protection ensures that bridge communications remain secure through the post-quantum transition, a guarantee no competitor offers. Existing bridges will require major security overhauls once quantum threat timelines compress, creating disruption and forced migrations.

2.5.4 Barriers to Competition

These advantages are protected by significant technical barriers:

Architecture depth: Full-stack PQC integration requires ground-up cryptographic architecture. Retrofitting post-quantum security onto an existing chain is a multi-year engineering effort that touches every protocol layer. The key difficulty lies in maintaining consensus safety while integrating larger signatures (4,595 bytes for ML-DSA-87 vs. 64 bytes for ECDSA), which requires redesigned block structures, bandwidth optimization, and signature aggregation logic. No existing L1 can cleanly adopt this without fundamental consensus restructuring.

State coordination complexity: Triple-VM coordination with unified state management is deeply integrated and cannot be achieved through modular add-ons or rollup strategies. Each VM must maintain atomicity guarantees across a shared ledger, requiring custom state transition functions, cross-VM call scheduling, and consistent ordering semantics. The technical challenge of synchronizing three independent execution models around a single global state, with atomic transactions spanning VM boundaries, creates a durable moat that cannot be replicated through bridge protocols or sharded architectures.

AI integration depth: Protocol-level AI optimization requires machine learning infrastructure woven into core consensus and execution paths, not bolted onto an

existing protocol as a smart contract or oracle service. QCAI's real-time threat detection operates within <1 millisecond per transaction, requiring specialized hardware integration and custom-built inference frameworks that process 50,000+ features per transaction. This cannot be retrofitted as an oracle; it must be part of the validator infrastructure from genesis.

PQC-native bridge operations: The Quantum-Certified Bridge (QCB) is the only cross-chain bridge protocol that authenticates messages with post-quantum cryptography, covering 25 directly connected L1 blockchains. Building PQC-authenticated bridges requires participating chains to upgrade their consensus security model, creating a first-mover advantage where early adoption chains become locked into QoreChain's ecosystem through bridge infrastructure dependencies.

Cryptographic continuity: Competitors face a catch-22: migrate to PQC late (after a CRQC emerges, suffering disruption) or migrate early (incurring engineering costs with uncertain ROI). QoreChain's established PQC foundation eliminates this choice for builders, making migration to quantum-safe infrastructure a matter of engineering economics rather than technological uncertainty.

2.6 Go-to-Market Strategy

2.6.1 Phase 1: Foundation (2026)

Q1-Q4 2026 focuses on establishing QoreChain as a production-grade blockchain with quantum-safe security and AI-native operations:

- Mainnet launch (Q2 2026), unified with the Token Generation Event, with full PQC, QCAI, and triple-VM capabilities, featuring a 200+ validator consensus set and targeting 5,000+ TPS design capacity. Pre-launch security audits by three independent firms (Halborn, Trail of Bits, and Least Authority) will validate cryptographic implementations and consensus safety before mainnet deployment.
- Validator onboarding program with cross-network validation licensing for 50+ initial validators. Validator rewards totaling 20 million QOR annually will incentivize participation; additional 3 million QOR rewards will be distributed to cross-network validators for bridge operations and state attestation services.
- Developer grants program distributing 15 million QOR to teams porting applications from Ethereum, Cosmos, and Solana ecosystems. Grants average 100 to 500 QOR per team depending on project scope, application type, and expected user impact. Priority focuses on DeFi protocols, NFT platforms, and governance systems.
- Light node network launch (SX and UX editions), targeting 10,000+ light node participants globally. Light node fund receives 3% of all protocol fees (estimated 25,000 QOR daily) distributed to participating nodes, creating immediate economic incentives for distributed participation.

2.6.2 Phase 2: Ecosystem Growth (2027)

Q1-Q4 2027 focuses on ecosystem maturation, developer tooling, and early enterprise adoption:

- QoreChain Studio general availability with 17-chain support (Ethereum, Arbitrum, Optimism, Polygon, Avalanche, BSC, Solana, Cardano, Polkadot, NEAR, TON, Starknet, Scroll, zkSync, Celo, Harmony, Fantom), enabling one-click smart contract generation, auditing, and deployment. Studio will process 500+ contract deployments monthly by year-end, supporting 2,000+ developers across all major blockchain ecosystems.

- DeFi protocol partnerships building quantum-safe DEX (targeting 200+ trading pairs and $50 million+ liquidity), lending platforms (supporting 15+ asset collateral types), and stablecoin protocols (USD, EUR, and emerging market currency pegs). Each protocol receives 1-3 million QOR implementation grants and ongoing liquidity incentives.

- Enterprise pilot programs with financial services (custody, settlement, treasury) and healthcare (medical records, drug supply chain) sectors, featuring regulatory guidance and compliance support. 5 to 8 pilot programs will begin operations with target settlements of 50,000 to 500,000 transactions monthly.

- Qore Wallet release in SX (desktop and mobile) and UX (browser) editions, supporting quantum-safe key management, cross-chain bridge operations, light node participation, and DeFi integration. Wallet targets 100,000+ users by year-end.

2.6.3 Phase 3: Multi-Chain Maturity (2027 to 2028)

2027-2028 consolidates QoreChain as the infrastructure layer for multi-chain applications and enterprise settlement:

- Full QCB deployment across 25 L1 blockchains (Ethereum, Solana, Cardano, Polkadot, NEAR, TON, and 19 others) with PQC-authenticated bridge operations supporting 500+ daily transfers and $500 million+ total value locked. Each connected L1 will operate 30+ licensed validators serving as bridge operators, watchers, and cross-network security providers. Bridge operations will achieve 99.9

- Cross-network validator ecosystem operating at scale with 500+ licensed validators across 25 L1s and 120+ IBC-linked networks. Validator participation in cross-network operations will generate 60% of average validator revenue alongside mainchain consensus rewards. Validator bond requirements will scale with network maturity, supporting 1,000-2,000 QOR minimum participation by year-end.

- Enterprise production deployments across regulated industries including financial settlement (custody, treasury, clearing), supply chain verification (provenance, regulatory compliance), and healthcare (medical records, insurance claims). 15-25 enterprise customers will reach production status with 5 to 50 million transactions annually across QoreChain and connected networks.

- Expansion of QCAI capabilities with on-chain model governance, enabling token holders to vote on algorithm parameters, threat detection thresholds, and AI model updates. QCAI will achieve 95

2.7 Investment Opportunity Summary

QoreChain occupies a unique position at the intersection of three converging macro trends: the quantum computing threat (creating urgency), the AI revolution (creating capability), and blockchain maturation (creating demand for next-generation infrastructure). No other project addresses all three simultaneously.

Market timing: The total addressable market spans quantum-safe enterprise infrastructure ($12 billion+), AI-enhanced blockchain services ($8 billion+), and cross-chain interoperability ($5 billion+), with QoreChain's Swiss-regulated foundation providing the institutional trust framework required for enterprise adoption. Current market timing is optimal: NIST has finalized PQC standards (ML-DSA and ML-KEM), making quantum-safe adoption feasible; major cloud providers have begun PQC migration; and enterprise blockchain adoption has matured beyond pilots to production deployments. The competitive window remains open for approximately 18 to 24 months before incumbent platforms begin PQC migrations, after which first-mover advantage consolidates.

Investment thesis: QoreChain captures value across three economic layers. First, as quantum computing timelines compress toward cryptographically relevant quantum computers (CRQCs), enterprises and governments face trillion-dollar infrastructure vulnerabilities. QoreChain's PQC-native design converts this risk into competitive advantage, positioning the platform as the settlement layer for quantum-safe digital assets. Second, AI-integrated consensus opens entirely new economic models: autonomous validators, dynamic fee markets optimized by machine learning, and security monitoring systems that detect threats at network speed. Third, the triple-VM execution environment reduces developer fragmentation by enabling atomic interoperability between Ethereum, Cosmos, and Solana ecosystems, capturing economic value from cross-ecosystem transactions currently lost to bridges and wrapped tokens.

Unique positioning: QoreChain's defensibility derives from three interlocking advantages. The protocol achieves quantum-safe security at genesis (not as an upgrade), eliminating the disruption and cost of later migration. The triple-VM architecture solves the ecosystem fragmentation problem that has limited blockchain adoption; developers can deploy to all three major ecosystems from a single platform. QCAI's protocol-level AI integration is fundamentally superior to oracle-based approaches, delivering sub-millisecond threat detection and dynamic optimization impossible in post-hoc systems. Together, these advantages create a platform effect: quantum-safe security attracts early adopters, the multi-VM capability attracts developers and liquidity providers, and QCAI's superior security metrics attract validators and institutional operators.

The strategic window is narrow. Organizations that establish quantum-safe infrastructure before CRQCs arrive will define the settlement layer for the post-quantum digital economy. Those that wait will face costly, disruptive migration under time pressure, loss of market position, and prolonged security uncertainty. QoreChain is designed to be the infrastructure they build on, and the time to establish market presence is now.

Chapter 3

Core Technology Architecture

QoreChain's architecture is a synthesis of three design imperatives: quantum-resistant security at every protocol layer, AI-native optimization as a first-class infrastructure component, and multi-ecosystem execution within a unified state environment. Built on the proven Cosmos SDK foundation and evolved significantly beyond it, the architecture addresses the fundamental limitations of current blockchain infrastructure while introducing capabilities that no existing platform provides in combination.

This chapter presents the complete technical stack, from cryptographic primitives through consensus and execution to networking and state management, with formal definitions, mathematical analysis, and developer-facing usage examples throughout.

3.1 Design Philosophy

Four governing principles shape every architectural decision in QoreChain:

Quantum-First Security. Post-quantum cryptography is not an add-on or migration target; it is a design constraint applied to every component. Consensus messages, state proofs, cross-chain packets, bridge operations, and smart contract signatures all use NIST-standardized PQC algorithms from genesis. Formally, for any protocol message m and signing key sk, the signature $\sigma = \text{Sign}_{\text{ML-DSA-87}}(sk, m)$ satisfies:

$$\Pr[\text{Forge}(\sigma') : \text{Verify}(pk, m', \sigma') = 1 \mid m' \neq m] \leq 2^{-\lambda} \tag{3.1}$$

where $\lambda \geq 128$ is the post-quantum security parameter, holding even against adversaries with bounded quantum computational resources. Every cryptographic operation is evaluated against the formal threat model of polynomial-time quantum adversaries, with no reliance on classical assumptions (discrete logarithm, factorization) that quantum computers render obsolete. This approach eliminates the migration uncertainty facing platforms that defer PQC adoption; quantum-safe guarantees are fundamental, not contingent on future upgrades.

Intelligence as Infrastructure. QCAI operates at the protocol layer, optimizing consensus, routing, and resource allocation continuously. Unlike external oracle or smart-contract-based AI, protocol-level integration enables sub-block-time decision cycles (millisecond-granularity decisions embedded within block production). The system executes ML models at core consensus and execution boundaries, enabling threat detection, validator selection, and load balancing to occur synchronously with transaction processing. This differs fundamentally from smart-contract-based AI that operates

with block-time latency (12-15 seconds); QoreChain's architecture eliminates this latency penalty, achieving real-time optimization impossible in post-hoc systems.

Multi-Ecosystem Compatibility. The triple-VM architecture (EVM, CosmWasm, SVM) ensures that developers from the Ethereum, Cosmos, and Solana ecosystems deploy contracts without rewriting. Cross-VM calls are atomic and execute within a single state transition, enabling sophisticated composability across three major blockchain execution models. The unified state environment means that account balances, contract state, and execution results are globally consistent across all three VMs; a transaction can atomically transfer funds from an EVM account, execute a CosmWasm contract, and settle the result in SVM within microseconds. This eliminates the fundamental limitation of bridge protocols that operate at transaction granularity with minutes of confirmation latency.

Modular Extensibility. Each functional domain (staking, bridging, licensing, AI services, light nodes, fee distribution) operates as an independent, upgradeable module. Modules interact through standardized interfaces defined in the protocol specification, enabling new modules to be introduced through governance proposals without modifying core consensus logic or requiring network forks. The module architecture supports hot-swapping implementations; a governance vote can authorize switching from one bridge implementation to another while maintaining all existing state and guarantees. This creates a framework for continuous protocol evolution driven by community governance, not centralized decision-making.

3.2 QoreChain Stack Overview

The QoreChain stack is organized into five layers, each responsible for a well-defined set of protocol functions. The layers interact through standardized interfaces, enabling independent evolution while maintaining system-wide invariants.

3.2.1 Layer 1: Cryptographic Foundation

The cryptographic layer provides quantum-resistant primitives consumed by every higher layer:

- **Digital Signatures**: ML-DSA-87 (FIPS 204, Dilithium-5) for all transaction signing, validator attestations, and governance votes. Security level: NIST Category 5 ($\geq$ 256-bit classical, $\geq$ 128-bit quantum).
- **Key Encapsulation**: ML-KEM-1024 (FIPS 203, Kyber) for key exchange in bridge communications, validator-to-validator channels, and encrypted state synchronization.
- **Hash Function**: SHAKE-256 (extendable-output function from SHA-3 family) for all Merkle tree construction, state proofs, and address derivation.
- **Stateless Signatures**: SLH-DSA (FIPS 205, SPHINCS+) available as a fallback signature scheme, providing hash-based security independent of lattice assumptions.
- **Hybrid Mode**: Classical ECDSA + ML-DSA-87 dual signatures for backward compatibility during ecosystem migration.

The cryptographic agility framework supports hot-swappable algorithm modules. Each algorithm is identified by an `AlgorithmID` registered in the on-chain algorithm registry. Transitions between algorithms are governed by a migration protocol:

$$\text{AlgTransition}(A_{\text{old}}, A_{\text{new}}, t_{\text{start}}, t_{\text{end}}) = \begin{cases} A_{\text{old}} & \text{if } t < t_{\text{start}} \\ A_{\text{old}} \cup A_{\text{new}} & \text{if } t_{\text{start}} \leq t \leq t_{\text{end}} \\ A_{\text{new}} & \text{if } t > t_{\text{end}} \end{cases} \tag{3.2}$$

During the overlap window $[t_{\text{start}}, t_{\text{end}}]$, both algorithms are accepted for verification, enabling zero-downtime migration.

Developer Usage: Signing a transaction with QoreChain's PQC primitives through the SDK:

```
// QoreChain SDK: Creating and signing a quantum-safe transaction
import { QoreChainClient, PQCKeyring } from "@qorechain/sdk";

// Generate a quantum-safe keypair (ML-DSA-87)
const keyring = await PQCKeyring.generate("ml-dsa-87");
console.log("Address:", keyring.address); // qor1abc...

// Create and sign a transfer transaction
const client = await QoreChainClient.connect("https://rpc.qorechain.io");
const tx = await client.buildTx({
  messages: [{
    typeUrl: "/qorechain.bank.v1.MsgSend",
    value: {
      fromAddress: keyring.address,
      toAddress: "qor1def...",
      amount: [{ denom: "uqor", amount: "1000000" }] // 1 QOR
    }
  }],
  fee: { amount: [{ denom: "uqor", amount: "1000" }], gas: "200000" }
});

// Sign with ML-DSA-87 (quantum-safe)
const signedTx = await keyring.signTransaction(tx);
const result = await client.broadcastTx(signedTx);
console.log("Tx hash:", result.transactionHash);
```

Hybrid mode for applications requiring backward compatibility:

```
// Hybrid signing: ECDSA + ML-DSA-87 dual signature
const hybridKeyring = await PQCKeyring.generate("hybrid-ecdsa-mldsa87");

// The resulting signature contains both classical and PQC components
// Verifiers accepting either scheme can validate the transaction
const hybridTx = await hybridKeyring.signTransaction(tx);
// hybridTx.signatures[0].algorithm = "hybrid-ecdsa-mldsa87"
// hybridTx.signatures[0].classicalSig = <ECDSA signature>
// hybridTx.signatures[0].pqcSig = <ML-DSA-87 signature>
```

3.2.2 Layer 2: Consensus

The **QoreChain Consensus Engine** implements a Combined Proof of Stake (CPoS) mechanism integrating three complementary selection strategies:

1. **Reputation Proof of Stake (RPoS)**: Validators accumulate reputation scores based on uptime, block production accuracy, and network contribution history.
2. **Delegated Proof of Stake (DPoS)**: Token holders delegate stake to validators, amplifying their selection weight.
3. **Traditional Proof of Stake (PoS)**: Direct stake provides baseline selection probability.

The combined validator selection weight is computed as:

$$W_i = \alpha \cdot S_i + \beta \cdot D_i + \gamma \cdot R_i \tag{3.3}$$

where S_i is validator i's self-bonded stake, D_i is total delegated stake, R_i is the reputation score normalized to $[0, 1]$, and $\alpha + \beta + \gamma = 1$ are governance-adjustable coefficients. The probability of validator i being selected as block proposer in round r follows:

$$P_i(r) = \frac{W_i}{\sum_{j \in \mathcal{V}} W_j} \cdot \mathrm{VRF}(sk_i, r) \tag{3.4}$$

where VRF is a verifiable random function providing unpredictable but verifiable proposer selection, and $\mathcal{V}$ is the active validator set.

Byzantine Fault Tolerance. The consensus protocol guarantees safety and liveness under the standard BFT assumption:

$$f < \frac{n}{3} \tag{3.5}$$

where n is the total number of validators and f is the number of Byzantine (malicious or faulty) validators. Agreement is achieved with probability at least $1 - 2^{-\lambda}$ where λ is the security parameter. All consensus messages (pre-votes, pre-commits, block proposals) are signed with ML-DSA-87.

QCAI-Enhanced Validator Selection. Beyond the deterministic weight formula, QCAI applies a reinforcement learning module that evaluates real-time network conditions to adjust consensus parameters dynamically:

$$\theta_{t+1} = \theta_t + \eta \cdot \nabla_\theta J(\theta_t) \tag{3.6}$$

where θ represents the consensus parameter vector (block size, timeout thresholds, validator set size), η is the learning rate, and $J(\theta)$ is the objective function balancing throughput, finality latency, and decentralization. The optimization runs continuously, adapting parameters between epochs.

Reputation Score Computation. A validator's reputation evolves over time according to:

$$R_i(t) = \lambda_R \cdot R_i(t-1) + (1 - \lambda_R) \cdot (\omega_u \cdot U_i(t) + \omega_p \cdot P_i(t) + \omega_v \cdot V_i(t)) \tag{3.7}$$

where $U_i(t)$ is uptime ratio, $P_i(t)$ is block production accuracy, $V_i(t)$ is vote participation rate, λ_R is the decay factor (preventing historical dominance), and $\omega_u+\omega_p+\omega_v = 1$ are contribution weights set by governance.

3.2.3 Layer 3: Execution

The execution layer provides a **triple-VM environment** operating within a single unified state:

- **EVM**: Full Solidity and Vyper compatibility. Ethereum contracts deploy without modification. Supports all EVM opcodes through the Shanghai upgrade, with quantum-safe extensions available through precompiled contracts.
- **CosmWasm VM**: WebAssembly-based contract execution supporting Rust, TypeScript, and Go. Provides fine-grained capability-based permissions and deterministic execution guarantees.
- **SVM**: BPF-based program execution compatible with Solana-native application patterns. Supports parallel transaction execution for non-conflicting state access.

Cross-VM Communication. Contracts in one VM can call contracts in another VM within a single atomic transaction. The cross-VM call protocol marshals data between VM-specific formats through a canonical intermediate representation:

$$\text{CrossVMCall}(C_{\text{src}}, C_{\text{dst}}, \text{data}) \rightarrow \text{Marshal}(V_{\text{src}}, V_{\text{dst}}, \text{data}) \rightarrow \text{Execute}(C_{\text{dst}}, \text{data}') \rightarrow \text{Unmarshal}(V_{\text{dst}}, V_{\text{src}}, \text{result}) \tag{3.8}$$

where C denotes a contract, V denotes a VM type, and the entire sequence is committed or reverted atomically.

Developer Usage: Deploying an EVM contract with quantum-safe PQC extensions:

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

import "@qorechain/pqc/IPQCVerifier.sol";

/// @title QuantumSafeVault
/// @notice A vault that requires ML-DSA-87 signature for withdrawals
contract QuantumSafeVault {
    IPQCVerifier public immutable pqcVerifier;
    mapping(address => uint256) public balances;
    mapping(address => bytes) public pqcPublicKeys;

    constructor(address _verifier){
        pqcVerifier = IPQCVerifier(_verifier);
    }

    /// @notice Register a PQC public key for quantum-safe withdrawals
    function registerPQCKey(bytes calldata mldsaPublicKey) external {
        require(mldsaPublicKey.length == 2592, "Invalid ML-DSA-87 key");
        pqcPublicKeys[msg.sender] = mldsaPublicKey;
    }

    /// @notice Deposit QOR into the vault
    function deposit() external payable {
        balances[msg.sender] += msg.value;
    }

    /// @notice Withdraw with quantum-safe signature verification
    function withdraw(
        uint256 amount,
        bytes calldata pqcSignature
    ) external {
        require(balances[msg.sender] >= amount, "Insufficient balance");

        // Verify ML-DSA-87 signature via precompiled contract
        bytes memory message = abi.encodePacked(
            msg.sender, amount, block.number
        );
        require(
            pqcVerifier.verifyMLDSA87(
                message, pqcSignature, pqcPublicKeys[msg.sender]
            ),
            "PQC signature verification failed"
        );

        balances[msg.sender] -= amount;
```

```
47         payable(msg.sender).transfer(amount);
48     }
49 }
```

Developer Usage: A CosmWasm contract calling an EVM contract through cross-VM communication:

```
use cosmwasm_std::{DepsMut, Env, MessageInfo, Response, Binary};
use qorechain_crossvm::{EvmCall, CrossVMExecute};

/// Execute a cross-VM call from CosmWasm to an EVM contract
pub fn execute_cross_vm_swap(
    deps: DepsMut,
    _env: Env,
    info: MessageInfo,
    evm_dex_address: String,
    amount_in: u128,
    min_amount_out: u128,
) -> Result<Response, ContractError> {
    // Encode the EVM function call (Uniswap-style swap)
    let calldata = EvmCall::new(&evm_dex_address)
        .function("swapExactTokensForTokens(uint256,uint256,address)")
        .param_uint256(amount_in)
        .param_uint256(min_amount_out)
        .param_address(&info.sender.to_string())
        .encode()?;

    // Execute atomically: if the EVM call reverts,
    // the entire CosmWasm transaction reverts
    let cross_vm_msg = CrossVMExecute::Evm(calldata);

    Ok(Response::new()
        .add_message(cross_vm_msg)
        .add_attribute("action", "cross_vm_swap")
        .add_attribute("vm_source", "cosmwasm")
        .add_attribute("vm_target", "evm"))
}
```

3.2.4 Layer 4: Application

The application layer comprises protocol modules and developer-facing services:

- **Protocol Modules**: Staking, governance, fee distribution, bridge (QCB), license registry, light node management, DEX, and QCAI services, each operating as an independent, upgradeable component.
- **QoreChain Studio**: 17-chain contract generation and AI-powered auditing via QCAI.
- **QoreChain Bridge (QCB)**: 25 direct L1 connections and more than 120 additional networks through integrated IBC.

Developer Usage: Querying QoreChain state through the SDK:

```
// Query validator set with reputation scores
const validators = await client.query.staking.validators({
  status: "BOND_STATUS_BONDED"
});

for (const v of validators){
  console.log(`${v.description.moniker}:`);
  console.log(` Stake: ${v.tokens} uqor`);
  console.log(` Reputation: ${v.reputationScore}`);
  console.log(` Delegators: ${v.delegatorCount}`);
  console.log(` License: ${v.crossNetworkLicense?.status}`);
}

// Query light node network status
const lightNodes = await client.query.lightnode.activeNodes({
  pagination: { limit: 100 }
});
console.log(`Active light nodes: ${lightNodes.total}`);
```

3.2.5 Layer 5: Network

The networking layer implements an **Intelligent Gossip Protocol (IGP)** enhanced with AI-driven peer selection and adaptive message routing. Traditional blockchain gossip protocols broadcast each message to all peers, creating $O(n^2)$ message complexity and high latency. Geographic distribution and heterogeneous network conditions (satellite, cellular, fixed broadband) require adaptive protocols that learn optimal routing paths rather than applying fixed topology assumptions. QoreChain's IGP applies QCAI to reduce this overhead through learned peer selection and dynamic routing tables that adapt to observed network conditions in real time:

- **Adaptive propagation**: QCAI predicts optimal message routing paths based on network topology (peer connections, geographic distribution), historical latency data (peer RTT measurements), and message priority (consensus messages, transactions, light node updates). The protocol maintains a probabilistic routing table that routes messages through peers expected to provide fastest propagation. This reduces message complexity from $O(n \log n)$ in traditional gossip (each peer forwards to $\log(n)$ peers) to near-optimal $O(n)$ in many network configurations. For example, a consensus message reaches 95% of peers in 3-5 hops instead of 8-12 hops, reducing latency from 5-10 seconds to 1-2 seconds.

- **Bandwidth management**: Message sizes and propagation frequency adapt to observed network conditions. During periods of high network congestion, the protocol increases message compression ratios and reduces transaction relay frequency (from microsecond-level relay to batched relay), preserving bandwidth for consensus-critical messages. During periods of low congestion, transmission is optimized for speed. The protocol maintains separate message queues for consensus (highest priority), state updates (medium priority), and transactions (lower priority), ensuring that network congestion never delays consensus.

- **Light node relay**: SX and UX light nodes participate in transaction relay and state validation, extending geographic coverage and reducing latency. Light nodes can relay transactions to nearby full nodes and other light nodes, creating geographically distributed relay networks that reduce the latency of transaction propagation. This is especially valuable in regions far from major validator clusters (e.g., Africa, South America, Southeast Asia), where light node gateways can relay transactions with lower latency than direct-to-validator relay.
- **Quantum-safe authentication**: All peer-to-peer communication is authenticated using ML-DSA-87 signatures on all messages. Peers sign their identity, message content, and timestamp, preventing message forgery and tampering. Where confidentiality is required (e.g., cross-network validator messages), encryption uses ML-KEM-1024 key encapsulation combined with AES-256, providing post-quantum security for sensitive network communications.

The IGP is designed to handle high-velocity networks (2,000-5,000 nodes) with extreme geographic distribution (global validators and light nodes) while maintaining sub-2-second consensus finality.

3.3 Multi-Layer Processing Architecture

QoreChain implements a hierarchical processing architecture inspired by CPU cache hierarchy principles: critical operations execute at the highest-security layer, while bulk operations are delegated to optimized subsidiary layers.

3.3.1 Architectural Overview

The architecture comprises four processing tiers:

1. **Main Chain**: Settlement, governance, and cross-chain coordination. Maximum security, deterministic finality.
2. **Sidechains**: Compute-intensive smart contract execution, AI inference, and complex cross-chain operations. Optimized for throughput.
3. **Paychains**: High-frequency micro-transactions and payment channels. Optimized for latency with batch settlement.
4. **Rollups**: Batched execution with on-chain settlement proofs. Four settlement modes (optimistic, ZK, based, sovereign) enable workload-specific tradeoffs between finality speed, trust assumptions, and cost. Managed through the Rollup Development Kit (RDK); see Chapter 7 for full treatment.

Each subsidiary chain maintains cryptographic commitments to its state, periodically anchored to the main chain through a **Hierarchical Commitment Scheme (HCS)**. For subsidiary chain C_k with state S_k at block height h:

$$\text{Commit}(C_k, h) = H_{\text{SHAKE-256}}(S_k^h \| h \| k \| \text{nonce}) \tag{3.9}$$

These commitments are included in main chain blocks at epoch boundaries, creating a verifiable chain of state anchors. The security guarantee is:

$$\Pr[\text{FraudAccepted}] \leq \frac{f_k}{n_k} \cdot 2^{-\lambda} \tag{3.10}$$

where f_k and n_k are the number of Byzantine and total validators on subsidiary chain k, and λ is the security parameter. Rollup layers use an alternative proof model (fraud proofs or validity proofs) detailed in Section 7.3.

3.3.2 Main Chain: Settlement and Governance

The main chain processes high-value transactions requiring maximum security guarantees (settlement, governance, treasury operations) and maintains quantum-safe Merkle state proofs for all subsidiary chains and rollups. It executes governance decisions (parameter changes, module upgrades, algorithm migrations, treasury allocation) with on-chain voting and deterministic finality. The main chain is designed to process approximately 500-1,000 transactions per second, scaling linearly with validator count and validator hardware.

State compression enables the main chain to validate subsidiary chain state without processing every individual transaction in those chains. Subsidiary chain state transitions are cryptographically aggregated before inclusion in main chain blocks, reducing the on-chain verification burden:

$$\text{AggState}(h) = \bigoplus_{k=1}^{K} \text{Commit}(C_k, h) \tag{3.11}$$

where $\bigoplus$ denotes the Merkle aggregation operator and K is the number of active subsidiary chains. Verification of a single subsidiary chain's state requires only $O(\log K)$ proof elements, enabling efficient validation of hundreds of subsidiary chains from a single main chain block.

The main chain also implements **governance without inflation**. Unlike many blockchain governance systems that mint new tokens for governance participation (creating long-term inflation), QoreChain's main chain uses QOR staking directly for voting power. Governance votes consume no additional tokens, use existing stake as voting weight, and create no inflation pressure. This aligns governance incentives with long-term protocol health (voters want protocol value to increase) and avoids the need for continual token emissions to incentivize participation.

Main chain blocks are produced every 2 seconds on average, with Byzantine fault tolerance under the standard f < n/3 assumption. Each block includes approximately 500-1000 settlement transactions, 5-10 governance proposals/votes, and aggregated state commitments from 20-50 active subsidiary chains.

3.3.3 Sidechains: Compute-Intensive Operations

Sidechains provide dedicated processing environments for:

- Smart contract execution requiring high computational resources
- QCAI inference and model evaluation tasks
- Complex cross-chain operations involving multiple bridge hops

Each sidechain operates an independent consensus instance with periodic state anchoring to the main chain. The anchoring interval Δ balances security (shorter intervals) against overhead (longer intervals):

$$\Delta^* = \arg\min_{\Delta} \left[\frac{C_{\text{anchor}}}{\Delta} + \mu \cdot \Delta \cdot P_{\text{fraud}} \right] \tag{3.12}$$

where C_{anchor} is the gas cost of anchoring, μ is the expected loss from fraud, and P_{fraud} is the probability of undetected fraud per block. QCAI dynamically adjusts Δ based on observed conditions.

3.3.4 Paychains: High-Frequency Transactions

Paychains are optimized for micro-transactions, payment channels, and high-frequency trading applications:

- Near-instant confirmation with probabilistic finality
- Batch settlement to the main chain at configurable intervals
- Deterministic finality achieved upon main chain settlement

Transaction batches are compressed using a rolling hash accumulator:

$$B_h = H(B_{h-1} \| tx_1 \| tx_2 \| \ldots \| tx_m) \tag{3.13}$$

where m is the batch size. The batch proof submitted to the main chain is $O(1)$ in size regardless of the number of transactions in the batch.

3.3.5 Rollups: Batched Execution with On-Chain Settlement

Rollups represent the fourth layer type in QoreChain's multi-layer architecture, introduced via the Rollup Development Kit (RDK). Rollups execute transactions off-chain in batches and submit compact settlement proofs to the main chain, inheriting main chain security while achieving higher throughput and lower per-transaction costs.

QoreChain supports four settlement modes, each offering distinct tradeoffs:

Settlement Mode	Proof Type	Finality	Trust Assumption	Cost
Optimistic	Fraud proof	7-day challenge window	1-of-N honest challenger	Low
ZK	Validity proof (SNARK/STARK)	Immediate on proof verification	Cryptographic soundness	High (proving)
Based	L1-sequenced	Main chain finality	Main chain validators	Medium
Sovereign	Self-attested	Rollup-defined	Rollup validator set	Lowest

Table 3.1: QoreChain rollup settlement modes

The state commitment for a rollup batch of m transactions is:

$$\text{BatchCommit}(b) = H_{\text{SHAKE-256}}\left(S_{\text{pre}} \| S_{\text{post}} \| \text{txRoot}(tx_1, \ldots, tx_m) \| \pi\right) \tag{3.14}$$

where S_{pre} and S_{post} are the pre-state and post-state roots, txRoot is the Merkle root of the transaction batch, and π is the proof (fraud proof, validity proof, or empty for sovereign mode). The full RDK specification, including rollup profiles, sequencer modes, data availability backends, and AI-optimized configuration, is detailed in Chapter 7.

3.3.6 Semantic Transaction Routing

QCAI classifies incoming transactions and routes them to the optimal processing tier, optimizing for the user's explicit or implicit preferences. The routing function $\mathcal{R}$ maps transaction features to a layer assignment:

$$\mathcal{R}(tx) = \arg\max_{l \in \{\text{main,side,pay,rollup}\}} [\omega_s \cdot \text{Security}(l, tx) + \omega_t \cdot \text{Throughput}(l) - \omega_c \cdot \text{Cost}(l, tx)] \tag{3.15}$$

where $\omega_s, \omega_t, \omega_c$ are learned weights balancing security requirements (learned from historical transaction selections), throughput availability (current load on each layer), and cost efficiency. The model learns these weights through reinforcement learning, observing user satisfaction with routing decisions (faster confirmation appreciated; cost overages disliked) and optimizing over time.

The model considers rich features: transaction value (higher value prefers main chain), complexity (estimated gas, prefers appropriate layer with sufficient capacity), state dependencies (cross-chain interactions require main chain coordination), urgency flags (DeFi liquidations prefer fastest route), and rollup affinity (transactions intended for a specific rollup are pre-routed).

Example routing decisions:

- **Governance vote**: Value-neutral, non-urgent governance transaction to main chain (requires main chain finality for immediate effect). Cost: 0.001 QOR. Confirmation time: 2-4 seconds (next block).
- **ERC-20 transfer**: Low-value (0.1 QOR), no urgency, non-critical, routed to paychain. Cost: 0.00001 QOR (100x cheaper). Confirmation time: 1 second (paychain), settlement to main chain within 1 hour.
- **Cross-chain DEX swap**: Medium-value, complexity bridges 2 chains, routed to sidechain + bridge. Cost: 0.01 QOR. Confirmation time: 5 seconds (sidechain) + 30 seconds (bridge), total 35 seconds.

Cross-Layer Fee Bundling. Users pay a single fee for a transaction regardless of which layers it touches. The protocol automatically distributes fees to validators and operators based on computational contribution:

$$\text{Fee}_i = \text{Fee}_{\text{total}} \cdot \frac{\text{Gas}_i}{\sum_j \text{Gas}_j} \tag{3.16}$$

where Gas_i is the gas consumed on layer i, and fees are automatically routed to the appropriate validators/operators/light nodes. This bundling prevents users from needing to understand multi-layer architecture or pay separate fees for each layer.

3.4 State Management

3.4.1 Quantum-Resistant State Proofs

All state is organized in a Merkle tree using SHAKE-256 as the hash function, providing quantum-resistant state authentication. For a state tree with N leaves, any state value v at path p can be verified with a proof of size $O(\log N)$:

$$\text{Verify}(\text{root}, p, v, \pi) = \begin{cases} \text{true} & \text{if } H^{(\log N)}(\pi, v) = \text{root} \\ \text{false} & \text{otherwise} \end{cases} \tag{3.17}$$

where π is the Merkle proof (sequence of sibling hashes) and $H^{(k)}$ denotes k successive applications of $H_{\text{SHAKE-256}}$.

Developer Usage: Verifying a state proof:

```
// Verify a quantum-safe state proof for an account balance
const proof = await client.query.auth.accountProof("qor1abc...");

const isValid = await client.verifyStateProof({
  root: latestBlock.appHash,     // SHAKE-256 state root
  key: proof.key,                 // Account path
  value: proof.value,             // Serialized account data
  proof: proof.merkleProof,       // Sibling hash chain
  hashFunction: "shake-256"       // PQC-resistant hash
});

console.log("Proof valid:", isValid); // true
console.log("Account balance:", decodeAccount(proof.value).balance);
```

3.4.2 Versioned State with Checkpointing

QoreChain maintains a versioned state database enabling efficient state queries at any historical block height h, a critical feature for light nodes and chain analysis. Rather than storing all historical state (which would require petabytes of storage), the protocol stores periodic checkpoints and diffs:

$$\text{State}(h) = \text{Checkpoint}(h_0) \oplus \bigoplus_{i=h_0+1}^{h} \Delta_i \tag{3.18}$$

where $\text{Checkpoint}(h_0)$ is the nearest preceding checkpoint (at epoch h_0, approximately every 12 hours or 21,600 blocks), and Δ_i is the state diff at block i (changes to balances, contract state, etc.). The state is recovered by starting with the checkpoint and replaying diffs up to the target height.

PQC-authenticated checkpoints at epoch boundaries provide verifiable state snapshots using ML-DSA-87 signatures from 2/3+ of validators. These checkpoints enable new nodes to synchronize from a recent checkpoint rather than replaying the full chain history. For example, a node joining the network today can download the latest checkpoint (verified by validator signatures), then download approximately 12 hours of diffs (roughly 1 GB of data), rather than downloading the full 5-year blockchain history (several terabytes).

This design also enables **historical state queries**: applications can request account state at any point in blockchain history (e.g., "what was Alice's balance at block 5,000,000?") without maintaining the full archive. The protocol stores checkpoint data indefinitely (on distributed storage) and keeps diffs for the most recent 6-12 months locally, providing a reasonable tradeoff between query capability and storage requirements.

Checkpoints are produced every epoch (approximately every 12 hours) and must be signed by at least 2/3 of the validator set, ensuring that historical state cannot be retroactively altered without consensus-level attack.

3.4.3 Elastic Consistency Model

Different transaction types operate under appropriate consistency guarantees:

- **Strong consistency**: High-value settlement transactions, governance votes, and bridge operations. Deterministic finality within one consensus round.
- **Eventual consistency**: Micro-transactions on paychains. Probabilistic finality with confirmation probability $p(t)$ increasing over time:

$$p(t) = 1 - e^{-\lambda_{\text{confirm}} \cdot t} \tag{3.19}$$

where λ_{confirm} is the confirmation rate parameter and t is time since submission. Full deterministic finality is achieved upon main chain settlement.

3.5 Module Architecture

QoreChain's functionality is organized into independent, upgradeable modules. Each module governs a specific protocol domain and communicates with other modules through standardized interfaces. Cross-module transactions are atomic: a bridge operation that triggers fee distribution and license verification executes as a single indivisible state transition.

3.5.1 Core Protocol Modules

New modules can be proposed and activated through governance without protocol forks. The module activation process follows:

$$\text{ModuleActivation} = \begin{cases} \text{Proposed} & \text{governance proposal submitted} \\ \text{Voting} & \text{quorum threshold: } q \geq \frac{1}{3} \text{ of bonded stake} \\ \text{Accepted} & \text{approval threshold: } v_{\text{yes}} > \frac{1}{2} \text{ of votes cast} \\ \text{Active} & \text{after upgrade height reached} \end{cases} \tag{3.20}$$

3.5.2 Fee Distribution Model

Every transaction fee F is distributed according to fixed allocation ratios:

$$F = 0.37 \cdot F_{\text{validators}} + 0.30 \cdot F_{\text{burn}} + 0.20 \cdot F_{\text{treasury}} + 0.10 \cdot F_{\text{stakers}} + 0.03 \cdot F_{\text{light\_nodes}} \tag{3.21}$$

The burn component creates sustained deflationary pressure. Let $S(t)$ denote the circulating supply at time t:

Module Domain	Responsibility
Staking & Delegation	Validator registration, stake management, delegation, epoch-based reward distribution
Governance	On-chain proposal submission, voting (weighted by stake and reputation), parameter change execution
Fee Distribution	37% validators / 30% burn / 20% treasury / 10% stakers / 3% light nodes
Bridge (QCB)	Cross-chain asset transfer and message passing across 25 L1s + 120+ IBC networks
License Registry	On-chain licensing for cross-network validator operations: watching, bridging, and validation of connected chains
Light Node Management	Registration, heartbeat liveness monitoring, stake-weighted reward distribution for SX and UX nodes
AI Services (QCAI)	Protocol-level AI operations: transaction routing, validator selection, anomaly detection, resource allocation
DEX	Automated market maker with liquidity pools, swap execution, and dynamic pricing
PQC	Post-quantum cryptographic operations exposed as protocol primitives and precompiled contracts

Table 3.2: QoreChain core protocol modules

$$S(t) = S(0) - \int_0^t 0.30 \cdot F(\tau)\, d\tau \tag{3.22}$$

where $F(\tau)$ is the aggregate fee at time τ. Since total supply is fixed at 4,500,000,000 QOR, the burn mechanism ensures monotonically decreasing circulating supply over the long term.

Validator rewards within the 37% allocation are distributed proportionally to contribution:

$$r_i = 0.37 \cdot F \cdot \frac{W_i}{\sum_{j \in \mathcal{V}} W_j} \tag{3.23}$$

where W_i is the combined weight of validator i (stake, delegation, and reputation as defined in Section 3.2.2).

3.6 Security Architecture

3.6.1 Defense in Depth

QoreChain implements four independent security layers. An attack must compromise multiple layers simultaneously to affect network state:

Layer 1, Cryptographic: ML-DSA-87, ML-KEM-1024, SHAKE-256 provide post-quantum security at NIST Category 5. The probability of forging a single signature is bounded by 2^{-128} even for a quantum adversary.

Layer 2, Consensus: BFT with $f < n/3$ tolerance. Combined reputation and stake weighting makes Sybil attacks economically infeasible. The cost of acquiring sufficient stake to control consensus is:

$$C_{\text{attack}} \geq \frac{1}{3} \cdot \sum_{i \in \mathcal{V}} S_i \cdot P_{\text{QOR}} \tag{3.24}$$

where P_{QOR} is the market price of QOR. As network value grows, the attack cost scales proportionally.

Layer 3, AI-Driven Monitoring: QCAI continuously analyzes transaction patterns, validator behavior, and network state. Anomaly detection models identify deviations from expected behavior distributions:

$$\text{Anomaly}(x) = \begin{cases} \text{true} & \text{if } \|x - \mu_{\text{window}}\| > k \cdot \sigma_{\text{window}} \\ \text{false} & \text{otherwise} \end{cases} \tag{3.25}$$

where μ_{window} and σ_{window} are the rolling mean and standard deviation of the monitored metric, and k is the sensitivity threshold (governance-adjustable).

Layer 4, Economic: Slashing for validator misbehavior (double-signing, extended downtime), license bond requirements for cross-network validation, and stake lockup periods create economic deterrence.

3.6.2 Cryptographic Agility

The protocol supports running multiple cryptographic algorithm versions simultaneously during migration windows (as defined in Section 3.2.1). Algorithm governance follows NIST guidance: when new standards are published or existing algorithms are deprecated, the community votes on upgrade parameters (A_{extnew}, $t_{extstart}$, t_{extend}) through the standard governance process. The protocol maintains an algorithm registry on-chain specifying active, phase-in, and deprecated algorithms. Validators must support all active algorithms and can choose to support phase-in algorithms to facilitate future migrations.

Migration framework: The algorithm migration process is designed to prevent disruption while maintaining security. When a new algorithm is proposed, governance specifies both the technical parameters (algorithm specification, security proofs) and the timeline. The phase-in period typically spans 2 to 4 weeks, allowing validators and light nodes to upgrade their software and cache new algorithm implementations. During this window, both old and new algorithms are accepted for signatures, creating redundancy that prevents consensus failure if a minority of validators fail to upgrade. The sunset period follows completion of phase-in, after which new transactions cannot use deprecated algorithms, though old signatures remain verifiable indefinitely for historical audit purposes.

Governance-controlled transitions: All algorithm transitions require explicit governance approval, preventing unilateral protocol changes. The governance process includes technical review (core developers and cryptography experts assess the proposed algorithm's security proofs and performance characteristics), economic review (token holders evaluate the cost of ecosystem migration), and timeline approval (validators confirm they can upgrade within the proposed window). This multi-stakeholder governance model ensures that cryptographic choices remain aligned with community

interests rather than driven by hidden cryptanalytic breakthroughs or isolated developer decisions.

Fallback strategies: If a subtle cryptanalytic flaw is discovered in an active algorithm after deployment, the protocol can execute an expedited governance vote to accelerate migration to a backup algorithm. For instance, if a new lattice attack surfaces, the community can vote to compress the timeline and shift to hash-based signing (SLH-DSA) within days rather than months, with hash-based signatures enabled as a phase-in algorithm on emergency notice.

For example, if a new lattice-based algorithm is standardized in 2028, the governance process would follow: (1) Proposal submission: Core developers propose upgrade parameters, including the new algorithm specification and suggested migration timeline; (2) Community voting: Token holders and validators vote on the proposal (requiring >50% approval and >= 33% quorum); (3) Phase-in period: Between t_{start} and t_{end}, both old and new algorithms are accepted for new signatures, enabling ecosystem migration; (4) Sunset: After t_{end}, the old algorithm is deprecated for new signatures but old signatures remain verifiable indefinitely.

This approach ensures that no algorithm is forced upon the network before consensus. If a competing algorithm emerges with superior properties, the community can vote to migrate. If a subtle cryptanalytic breakthrough affects one algorithm family (e.g., lattice problems), the community can vote to accelerate migration to hash-based alternatives.

3.6.3 AI-Enhanced Threat Detection

QCAI's security monitoring operates at three time scales, enabling multi-layer threat detection that identifies attacks from minutes (transaction-level) to months (epoch-level aggregate attacks). This multi-timescale approach detects both fast attacks (transaction censorship, double-spending attempts) and slow attacks (gradual validator set concentration, stake accumulation by single entities). Early detection enables proactive governance response before threats compromise network security or validator incentives.

- **Real-time** (per-transaction): Anomaly scoring for individual transactions before inclusion in blocks. The system monitors transaction patterns (gas usage, value distribution, recipient patterns) and flags deviations from learned baseline distributions. Suspicious transactions (e.g., high-value transfers to newly created accounts, unusual cross-chain bridge patterns) are routed through additional verification or marked for validator review. The system analyzes approximately 50,000 features per transaction (gas usage, recipient reputation, value deviation from historical average, temporal clustering) in less than 1 millisecond using specialized ML hardware. Real-time detection prevents obvious attack patterns from propagating through the network.

- **Block-level** (per-block): Statistical analysis of block composition, fee patterns, and validator behavior across the most recent block window (last 50 blocks, approximately 50 minutes). The system monitors block producer (validator) consistency, transaction ordering patterns, and potential collusion signals (e.g., related validators repeatedly producing blocks with unusual transaction ordering that benefits colluding accounts). It detects censorship attempts (repeated rejection

of specific transactions), eclipse attacks (sustained communication isolation from specific peers), and double-signing attempts. Metrics are aggregated per validator and compared against reputation scores; validators with declining metrics face reputation penalties. Block-level detection identifies coordination patterns that individual transactions might not reveal.

- **Epoch-level** (per-epoch, approximately 12 hours): Deep analysis of network trends, stake distribution changes, governance activity, and cross-chain traffic patterns. The system monitors for slow-moving attacks (e.g., gradual accumulation of stake by a single entity, gradual increase in validator set concentration) and systemic risks (e.g., unusual cross-chain traffic patterns suggesting bridge-based attacks, sustained network latency affecting light nodes). It analyzes governance proposal patterns to identify coordinated voting blocs or governance spam. Epoch-level analysis triggers governance recommendations and allows the community to vote on parameter adjustments before threats manifest. This tier detects attacks requiring weeks or months to execute.

  All three time scales feed into a unified anomaly scoring system that produces a network-wide security health score (0-100) displayed to validators and the community. Sustained scores below 80 trigger governance recommendations for protocol parameter adjustments or validator set rebalancing. This creates a continuous loop where security metrics drive governance actions before threats escalate.

3.7 Performance Design Targets

Metric	Design Target
Transaction throughput (main chain)	5,000+ TPS
Finality	Sub-second (designed)
Block time	1 to 2 seconds (configurable)
Cross-VM call overhead	Negligible (same-state execution)
Light node sync time	Minutes (from nearest checkpoint)
ML-DSA-87 signature verification	< 1 ms per operation
ML-KEM-1024 encapsulation	< 0.5 ms per operation
SHAKE-256 hashing	> 500 MB/s
State proof verification	$O(\log N)$
Paychain batch settlement	Configurable (100 ms to 10 s)

Table 3.3: QoreChain performance design targets. Multi-node testnet benchmarks will be published as validation proceeds.

The theoretical maximum throughput of the multi-layer architecture, accounting for all three processing tiers, is:

$$\text{TPS}_{\text{max}} = \text{TPS}_{\text{main}} + \sum_{k=1}^{K_s} \text{TPS}_{\text{side},k} + \sum_{k=1}^{K_p} \text{TPS}_{\text{pay},k} \tag{3.26}$$

where K_s and K_p are the number of active sidechains and paychains respectively. As the network scales horizontally by adding subsidiary chains, aggregate throughput increases linearly.

Chapter 4

Post-Quantum Cryptography

Post-quantum cryptography is QoreChain's first design constraint. Every protocol component, from transaction signing to consensus messaging, from state proofs to cross-chain packet authentication, uses NIST-standardized post-quantum algorithms from genesis. This chapter presents the mathematical foundations, algorithm specifications, integration points across the protocol stack, the hybrid migration framework, and the cryptographic agility architecture that ensures QoreChain remains secure as both quantum hardware and PQC standards evolve.

4.1 Theoretical Foundations

4.1.1 The Quantum Threat to Classical Cryptography

The security of virtually all deployed blockchain cryptography rests on two computational hardness assumptions: the integer factorization problem (RSA) and the elliptic curve discrete logarithm problem (ECDLP). Both are efficiently solvable by a quantum computer running Shor's algorithm.

Shor's Algorithm. For an n-bit integer N, Shor's algorithm finds a factor in time $O(n^2 \log n \log\log n)$ on a quantum computer, compared to the best known classical algorithm (general number field sieve) requiring sub-exponential time $L_N[1/3, (64/9)^{1/3}]$. For elliptic curve cryptography, the quantum speedup is even more dramatic:

$$T_{\text{classical}}(\text{ECDLP}) = O(\sqrt{p}) \quad \text{vs.} \quad T_{\text{quantum}}(\text{ECDLP}) = O(\log^3 p) \tag{4.1}$$

where p is the order of the elliptic curve group. For the secp256k1 curve used by Bitcoin and Ethereum, $p \approx 2^{256}$, yielding classical security of approximately 2^{128} operations but quantum security of only $O(\log^3 2^{256}) = O(256^3) \approx 2^{24}$ operations, a trivial computation.

Grover's Algorithm. For symmetric cryptography and hash functions, Grover's algorithm provides a quadratic speedup for unstructured search:

$$T_{\text{Grover}}(N) = O(\sqrt{N}) \tag{4.2}$$

This effectively halves the security level of symmetric primitives: AES-256 provides 128-bit quantum security, and a 256-bit hash function provides 128-bit collision resistance against quantum adversaries. QoreChain's use of SHAKE-256 (256-bit output) therefore provides $\geq$ 128-bit quantum security for all hash-based operations.

Qubit Requirements. Recent estimates indicate that breaking ECDSA-256 requires approximately 2,330 logical qubits with a circuit depth of $O(10^9)$ gates. With error correction overhead (current estimates: 1,000 to 10,000 physical qubits per logical qubit), a cryptographically relevant quantum computer requires between 2.3×10^6 and 2.3×10^7 physical qubits. Current hardware (Google Willow: 105 qubits; IBM roadmap: ∼200 logical qubits by 2029) is progressing toward this threshold.

4.1.2 Lattice-Based Cryptography

QoreChain's primary PQC algorithms (ML-DSA-87 and ML-KEM-1024) are built on lattice problems, specifically the Module Learning With Errors (MLWE) and Module Short Integer Solution (MSIS) problems.

Definition (Module-LWE). Let $R_q = \mathbb{Z}_q[X]/(X^n + 1)$ be a polynomial ring. The MLWE problem is: given a uniformly random matrix $\mathbf{A} \in R_q^{k \times k}$ and a vector $\mathbf{b} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e} \pmod q$, where $\mathbf{s}, \mathbf{e} \leftarrow \chi_\eta$ are sampled from a centered binomial distribution with parameter η, distinguish $(\mathbf{A}, \mathbf{b})$ from $(\mathbf{A}, \mathbf{u})$ where $\mathbf{u}$ is uniformly random.

$$\text{MLWE}_{n,k,q,\eta}: \quad \text{Adv}[\mathcal{A}] = |\Pr[\mathcal{A}(\mathbf{A}, \mathbf{As} + \mathbf{e}) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{u}) = 1]| \leq \text{negl}(\lambda) \tag{4.3}$$

Definition (Module-SIS). Given $\mathbf{A} \in R_q^{k \times \ell}$, find a short non-zero vector $\mathbf{z} \in R^\ell$ such that $\mathbf{A} \cdot \mathbf{z} = \mathbf{0} \pmod q$ and $\|\mathbf{z}\| \leq \beta$.

$$\text{MSIS}_{n,k,\ell,q,\beta}: \quad \Pr\left[\mathcal{A}(\mathbf{A}) = \mathbf{z} : \mathbf{Az} = \mathbf{0},\ 0 < \|\mathbf{z}\| \leq \beta\right] \leq \text{negl}(\lambda) \tag{4.4}$$

Security Reduction. The hardness of MLWE and MSIS reduces to worst-case lattice problems. Specifically, solving MLWE with non-negligible advantage implies an efficient algorithm for the Shortest Vector Problem (SVP) on module lattices:

$$\text{MLWE} \leq_p \text{Module-SVP}_\gamma \quad \text{for } \gamma = \text{poly}(n, k, q) \tag{4.5}$$

The best known quantum algorithm for SVP achieves only a polynomial speedup over classical methods, meaning lattice problems remain hard even for quantum adversaries. This is the fundamental reason QoreChain standardizes on lattice-based cryptography.

4.1.3 Hash-Based Cryptography

SHAKE-256 belongs to the SHA-3 (Keccak) family and is an extendable-output function (XOF) based on the sponge construction:

$$\text{SHAKE-256}(M) = \text{Sponge}[f, \text{pad}, r](M, d) \tag{4.6}$$

where f is the Keccak-$f[1600]$ permutation, $r = 1088$ is the rate, and d is the desired output length. The security against collision attacks is:

$$\text{Collision resistance} = \min\left(\frac{d}{2}, \frac{c}{2}\right) \text{ bits} \tag{4.7}$$

where $c = 1600 - r = 512$ is the capacity. For $d = 256$, this yields 128-bit collision resistance, which is also the quantum collision resistance (Grover provides at most a cubic-root speedup for collision finding via the BHT algorithm).

SLH-DSA (FIPS 205, SPHINCS+) provides a signature scheme based solely on hash function security, independent of any algebraic structure. Its security relies on three properties of the underlying hash function:

1. Second pre-image resistance of the hash function family
2. Unforgeability of the WOTS+ one-time signature scheme
3. Collision resistance of the hypertree construction

SLH-DSA serves as QoreChain's fallback signature scheme: if lattice assumptions are ever weakened by mathematical or quantum advances, hash-based signatures provide a conservative alternative with well-understood security guarantees.

4.2 QoreChain's PQC Algorithm Suite

4.2.1 ML-DSA-87 (FIPS 204, Dilithium-5)

ML-DSA-87 is QoreChain's primary signature algorithm, applied to all transaction signing, consensus messaging, governance votes, bridge attestations, and validator credentials.

Parameters:

Parameter	Value
Security level	NIST Category 5 ($\geq 2^{256}$ classical, $\geq 2^{128}$ quantum)
Module rank (k, ℓ)	$k = 8$, $\ell = 7$
Modulus (q)	8,380,417
Public key size	2,592 bytes
Private key size	4,864 bytes
Signature size	4,595 bytes
Signing time	$\approx$ 1.5 ms (commodity CPU)
Verification time	$\approx$ 0.5 ms (commodity CPU)

Table 4.1: ML-DSA-87 parameter specifications

Signing Algorithm. The signer holds a secret key $sk = (\rho, K, \mathrm{tr}, \mathbf{s}_1, \mathbf{s}_2, \hat{\mathbf{t}}_0)$ and produces a signature on message M as follows:

1. Compute the message representative: $\mu = H(\mathrm{tr} \| M)$
2. Sample a masking vector: $\mathbf{y} \leftarrow S^{\ell}_{\gamma_1 - 1}$
3. Compute the commitment: $\mathbf{w} = \mathbf{A} \cdot \mathbf{y}$, decompose into high and low bits: $\mathbf{w}_1 = \mathrm{HighBits}(\mathbf{w})$
4. Compute the challenge: $\tilde{c} = H(\mu \| \mathbf{w}_1)$, expand to sparse polynomial $c \in R_q$ with $\|c\|_1 = \tau$
5. Compute the response: $\mathbf{z} = \mathbf{y} + c \cdot \mathbf{s}_1$

6. Rejection sampling: if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\|\text{LowBits}(\mathbf{w} - c \cdot \mathbf{s}_2)\|_\infty \geq \gamma_2 - \beta$, restart from step 2

7. Output signature $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$ where $\mathbf{h}$ encodes hint bits

Verification. Given public key $pk = (\rho, \mathbf{t}_1)$, message M, and signature $\sigma = (\tilde{c}, \mathbf{z}, \mathbf{h})$:

1. Reconstruct $\mathbf{A}$ from ρ

2. Expand challenge: $c = \text{SampleInBall}(\tilde{c})$

3. Compute: $\mathbf{w}'_1 = \text{UseHint}(\mathbf{h}, \mathbf{Az} - c \cdot \mathbf{t}_1 \cdot 2^d)$

4. Accept if $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ and $\tilde{c} = H(\mu \| \mathbf{w}'_1)$

The security proof shows that forging a signature requires solving the MSIS problem:

$$\text{Adv}^{\text{EU-CMA}}_{\text{ML-DSA-87}}(\mathcal{A}) \leq \text{Adv}^{\text{MSIS}} n, k + \ell, q, \beta'(\mathcal{B}) + \text{Adv}^{\text{MLWE}} n, k, q, \eta(\mathcal{B}') + \frac{q_H}{2^\lambda} \tag{4.8}$$

where q_H is the number of hash queries and λ is the security parameter.

Developer Usage: Signing and verifying with ML-DSA-87 via the QoreChain CLI:

```
# Generate a quantum-safe keypair
$ qorechaind keys add my-wallet --algo ml-dsa-87
- address: qor1x7kq4m9p3rz...
  name: my-wallet
  pubkey: '{"@type":"/qorechain.crypto.mldsa87.PubKey","key":"Aqr5..."}'
  type: local

# Sign a transaction
$ qorechaind tx bank send my-wallet qor1abc... 1000000uqor \
    --chain-id qorechain-diana \
    --fees 1000uqor \
    --sign-mode ml-dsa-87

# Verify a signature offline
$ qorechaind verify-signature \
    --pubkey "Aqr5..." \
    --signature "BKZN..." \
    --message tx_bytes.bin
Signature valid: true (algorithm: ML-DSA-87, security: NIST Category 5)
```

4.2.2 ML-KEM-1024 (FIPS 203, Kyber)

ML-KEM-1024 provides key encapsulation for secure channel establishment between validators, bridge relayers, and light nodes.

Parameters:

Key Generation. Generate $(\mathbf{A}, \mathbf{t}) \in R_q^{k \times k} \times R_q^k$ where $\mathbf{t} = \mathbf{A} \cdot \mathbf{s} + \mathbf{e}$ with $\mathbf{s}, \mathbf{e} \leftarrow \chi_\eta$:

Parameter	Value
Security level	NIST Category 5
Module rank (k)	4
Modulus (q)	3,329
Public key size	1,568 bytes
Ciphertext size	1,568 bytes
Shared secret size	32 bytes
Encapsulation time	≈ 0.3 ms
Decapsulation time	≈ 0.4 ms

Table 4.2: ML-KEM-1024 parameter specifications

$$\text{KeyGen}() \to (pk = (\mathbf{A}, \mathbf{t}),\ sk = \mathbf{s}) \tag{4.9}$$

Encapsulation. The sender computes a shared secret K and ciphertext $\mathbf{c}$:

$$\mathbf{r} \leftarrow \chi_\eta, \quad \mathbf{e}_1 \leftarrow \chi_\eta, \quad e_2 \leftarrow \chi_\eta \tag{4.10}$$
$$\mathbf{u} = \mathbf{A}^T \cdot \mathbf{r} + \mathbf{e}_1 \tag{4.11}$$
$$v = \mathbf{t}^T \cdot \mathbf{r} + e_2 + \lceil q/2 \rfloor \cdot m \tag{4.12}$$
$$\mathbf{c} = (\text{Compress}(\mathbf{u}), \text{Compress}(v)) \tag{4.13}$$
$$K = H(m \| H(\mathbf{c})) \tag{4.14}$$

where m is a random message used to derive the shared secret.

Decapsulation. The receiver recovers K using the secret key:

$$m' = \text{Decompress}(v) - \mathbf{s}^T \cdot \text{Decompress}(\mathbf{u}) \tag{4.15}$$

followed by re-encryption verification (Fujisaki-Okamoto transform) to achieve IND-CCA2 security:

$$\text{Adv}^{\text{IND-CCA2}}_{\text{ML-KEM-1024}}(\mathcal{A}) \leq \text{Adv}^{\text{MLWE}} n, k, q, \eta(\mathcal{B}) + \frac{q_D}{2^\lambda} \tag{4.16}$$

where q_D is the number of decapsulation queries.

Developer Usage: Establishing a quantum-safe channel between two nodes:

```
// QoreChain SDK: Quantum-safe key exchange for secure messaging
import { PQCKeyExchange } from "@qorechain/sdk";

// Node A generates an ML-KEM-1024 keypair
const nodeA = await PQCKeyExchange.generateKeypair("ml-kem-1024");

// Node B encapsulates a shared secret using Node A's public key
const { ciphertext, sharedSecret: secretB } =
    await PQCKeyExchange.encapsulate(nodeA.publicKey);

// Node A decapsulates to obtain the same shared secret
const secretA = await PQCKeyExchange.decapsulate(
    nodeA.privateKey, ciphertext
```

```
);

// Both secrets are identical (32 bytes)
console.log("Secrets match:", secretA.equals(secretB)); // true

// Use the shared secret for AES-256-GCM encrypted communication
const channel = await PQCKeyExchange.createSecureChannel(secretA);
await channel.send("Quantum-safe encrypted message");
```

4.2.3 SLH-DSA (FIPS 205, SPHINCS+)

SLH-DSA is available as a fallback signature scheme whose security depends solely on the collision resistance and second pre-image resistance of the underlying hash function, independent of any algebraic structure.

Parameter	SLH-DSA-SHAKE-256f	SLH-DSA-SHAKE-256s
Security level	NIST Category 5	NIST Category 5
Public key size	64 bytes	64 bytes
Signature size	49,856 bytes	29,792 bytes
Signing time	$\approx$ 3 ms (fast)	$\approx$ 150 ms (small)
Verification time	$\approx$ 2 ms	$\approx$ 5 ms

Table 4.3: SLH-DSA parameter specifications (fast vs. small variants)

The significantly larger signature size (49,856 bytes vs. 4,595 for ML-DSA-87) makes SLH-DSA impractical as a default, but its assumption-independent security makes it valuable as a conservative backup. QoreChain's cryptographic agility framework allows governance to activate SLH-DSA as the primary scheme if lattice assumptions are compromised.

4.2.4 SHAKE-256: Hash Foundation

SHAKE-256, a variable-output-length extendable-output function (XOF) from the SHA-3 family, is used for all hash operations throughout the protocol. SHAKE-256 provides flexible output lengths (32 bytes to 64 bytes or more), enabling efficient Merkle tree construction, address derivation, and commitment schemes without requiring separate hash functions. The XOF property allows different parts of the protocol to extract different amounts of hash output from the same input, reducing computational redundancy. SHAKE-256 is a NIST-standardized primitive with no known quantum attacks better than exhaustive search, providing 256-bit classical and post-quantum security across 2030-2050 timescales.

- **Merkle tree construction**: All state proofs use SHAKE-256 internal nodes
- **Address derivation**: addr = Bech32("qor", SHAKE-256(pk)[0..20])
- **Commitment schemes**: Transaction commitments, block commitments, and cross-chain packet commitments

- **VRF construction**: Verifiable random function for proposer selection uses SHAKE-256 as the underlying hash

4.3 PQC Integration Across the Protocol Stack

QoreChain applies PQC at every layer where cryptographic operations occur. There is no protocol path where a classical-only cryptographic assumption is the sole security guarantee.

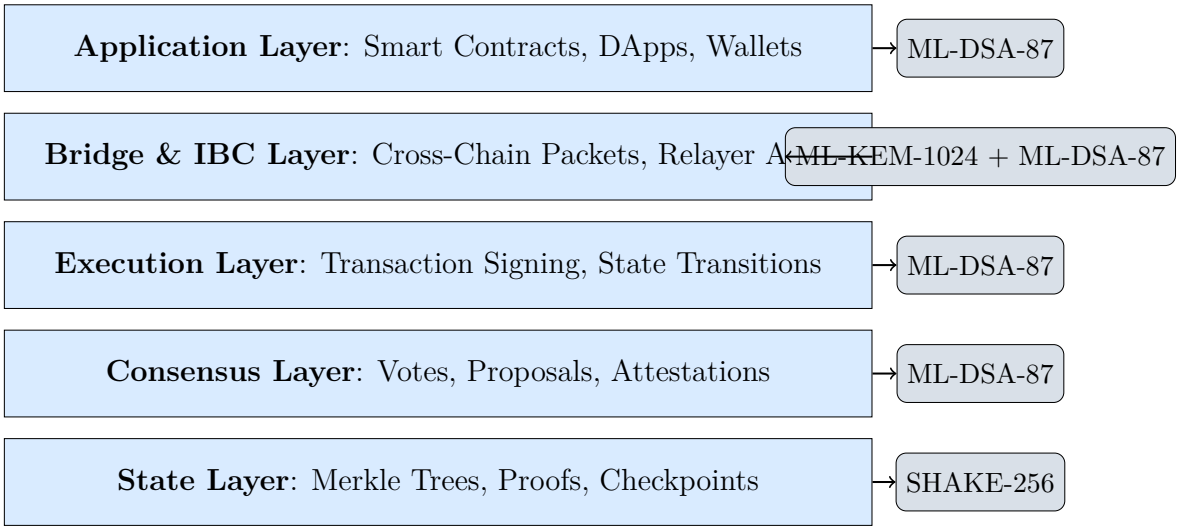

Figure 4.1: PQC algorithm assignment across protocol layers

4.3.1 Transaction Signing and Verification

Every transaction broadcast to the QoreChain network carries an ML-DSA-87 signature. Verification occurs at two checkpoints:

1. **Mempool entry**: Nodes verify the signature before accepting the transaction into the mempool. Invalid signatures are rejected immediately, preventing resource exhaustion attacks.

2. **Block validation**: Validators re-verify all transaction signatures during block proposal validation, ensuring consensus on a set of validly signed transactions.

The total verification cost per block with m transactions is:

$$T_{\text{verify}}^{\text{block}} = m \cdot T_{\text{verify}}^{\text{ML-DSA-87}} \approx m \cdot 0.5 \text{ ms} \tag{4.17}$$

For a block containing 5,000 transactions, total verification time is approximately 2.5 seconds in sequential mode. Parallel verification across p cores reduces this to:

$$T_{\text{verify}}^{\text{parallel}} = \frac{m \cdot T_{\text{verify}}^{\text{ML-DSA-87}}}{p} \approx \frac{2{,}500 \text{ ms}}{p} \tag{4.18}$$

On an 8-core commodity server, parallel verification completes in approximately 312 ms, well within the target block time.

4.3.2 Consensus Messages

All consensus protocol messages are ML-DSA-87 signed:

- **Block proposals**: The proposer signs the block header, transaction root, and state root
- **Pre-votes**: Each validator signs a vote for the proposed block
- **Pre-commits**: Each validator signs a commitment to the agreed block
- **VRF proofs**: Proposer selection proofs use PQC-based verifiable random functions

For a validator set of size n, the total signature data per consensus round is:

$$D_{\text{consensus}} = (1 + 2n) \cdot |\sigma_{\text{ML-DSA-87}}| = (1 + 2n) \cdot 4{,}595 \text{ bytes} \tag{4.19}$$

With $n = 100$ validators, this is approximately 924 KB per round, a manageable overhead for modern network connections.

4.3.3 State Proofs

All Merkle trees use SHAKE-256 as the hash function. For a state tree with N key-value pairs, a membership proof requires $\lceil \log_2 N \rceil$ sibling hashes:

$$|\pi_{\text{membership}}| = \lceil \log_2 N \rceil \cdot 32 \text{ bytes} \tag{4.20}$$

For $N = 10^8$ state entries (a mature blockchain), a proof is $\lceil \log_2 10^8 \rceil \cdot 32 = 27 \cdot 32 = 864$ bytes, enabling efficient verification by light clients and mobile devices.

Developer Usage: Querying and verifying a state proof from a light client:

```
// Light client: verify account balance without trusting a full node
import { LightClient, MerkleVerifier } from "@qorechain/sdk";

const light = await LightClient.connect("https://rpc.qorechain.io");

// Fetch the latest trusted block header (PQC-signed by 2/3+ validators)
const header = await light.getLatestTrustedHeader();

// Request account state with Merkle proof
const { value, proof } = await light.queryWithProof(
  "/store/acc/key",
  Buffer.from("qor1abc...")
);

// Verify the proof locally using SHAKE-256 Merkle verification
const valid = MerkleVerifier.verify({
  root: header.appHash,
  path: "/store/acc/key",
  value: value,
  proof: proof,
  hashFunction: "shake-256"
```

```
});

console.log("Balance verified:", valid);
// No trust in the full node required; only trust in 2/3+ validator sigs
```

4.3.4 Cross-Chain Communications

QoreChain Bridge (QCB) and IBC packets use dual-layer PQC protection:

1. **Confidentiality**: ML-KEM-1024 establishes a shared secret between the sending and receiving relay nodes. The packet payload is encrypted with AES-256-GCM using the derived key.

2. **Authenticity**: The encrypted packet is signed with ML-DSA-87 by the sending validator. Receiving validators verify the signature before decryption.

The security of a cross-chain packet P is formalized as:

$$\text{Security}(P) = \text{IND-CCA2}(\text{ML-KEM-1024}) \wedge \text{EU-CMA}(\text{ML-DSA-87}) \tag{4.21}$$

An adversary must break both the key encapsulation and the signature scheme to forge or read a cross-chain message, requiring solutions to both MLWE and MSIS simultaneously.

Developer Usage: Sending a PQC-secured cross-chain transfer:

```
// Initiate a quantum-safe cross-chain transfer via QCB
const bridge = await client.bridge.connect();

const transfer = await bridge.send({
  sourceChain: "qorechain",
  destinationChain: "ethereum",
  token: { denom: "uqor", amount: "5000000" }, // 5 QOR
  receiver: "0x742d35Cc6634C0532925a3b844Bc9e7595f2bD18",
  security: "pqc-full", // ML-KEM-1024 + ML-DSA-87
  timeout: { revisionHeight: 1000 }
});

console.log("Bridge transfer ID:", transfer.packetId);
console.log("PQC encryption:", transfer.encryption); // "ML-KEM-1024"
console.log("PQC signature:", transfer.signatureAlgo); // "ML-DSA-87"
```

4.3.5 Validator Key Management

Validators generate and manage two PQC keypairs:

- **Consensus key** (ML-DSA-87): Used for block proposals, pre-votes, and pre-commits

- **Transport key** (ML-KEM-1024): Used for establishing encrypted peer-to-peer channels

Both keys are generated within Hardware Security Modules (HSMs) where available, and stored with AES-256-GCM encryption at rest. Key rotation follows a governance-defined schedule, with on-chain announcement of new public keys and a grace period for verification convergence.

```
# Validator key generation and registration
$ qorechaind init my-validator --chain-id qorechain-diana

# Generate PQC consensus key
$ qorechaind keys add validator-consensus \
    --algo ml-dsa-87 --keyring-backend file

# Generate PQC transport key
$ qorechaind keys add validator-transport \
    --algo ml-kem-1024 --keyring-backend file

# Register validator with PQC keys
$ qorechaind tx staking create-validator \
    --amount 1000000000uqor \
    --pubkey $(qorechaind tendermint show-validator) \
    --moniker "my-validator" \
    --commission-rate 0.05 \
    --pqc-consensus-key validator-consensus \
    --pqc-transport-key validator-transport
```

4.3.6 Smart Contract PQC Primitives

PQC operations are exposed to smart contract developers through precompiled contracts (EVM) and native SDK functions (CosmWasm), enabling applications to perform quantum-safe cryptographic operations within on-chain logic.

EVM Precompiled Contracts:

Address	Function	Gas Cost
0x0...0100	ML-DSA-87 Verify(msg, σ, pk)	10,000 gas
0x0...0101	ML-KEM-1024 Encapsulate(pk)	8,000 gas
0x0...0102	ML-KEM-1024 Decapsulate(sk, ct)	8,000 gas
0x0...0103	SHAKE-256 Hash(data,outLen)	60 + 12/word gas
0x0...0104	SLH-DSA Verify(msg, σ, pk)	25,000 gas

Table 4.4: PQC precompiled contract addresses and gas costs

Developer Usage: Calling PQC precompiled contracts from Solidity:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

```

```
/// @title PQC-Protected Multisig
/// @notice Requires ML-DSA-87 signatures from M of N signers
contract QuantumMultisig {
    address constant PQC_VERIFY = address(0x0100);

    bytes[] public signerPubKeys;
    uint256 public threshold;

    constructor(bytes[] memory _pubKeys, uint256 _threshold){
        require(_threshold <= _pubKeys.length, "Invalid threshold");
        signerPubKeys = _pubKeys;
        threshold = _threshold;
    }

    /// @notice Execute a transaction with M-of-N PQC signatures
    function execute(
        address target,
        uint256 value,
        bytes calldata data,
        bytes[] calldata signatures
    ) external returns (bytes memory){
        require(signatures.length >= threshold, "Not enough sigs");

        bytes32 txHash = keccak256(
            abi.encodePacked(target, value, data, block.number)
        );

        uint256 validCount = 0;
        for (uint256 i = 0; i < signatures.length; i++){
            // Call ML-DSA-87 verify precompile
            (bool success, bytes memory result) =
                PQC_VERIFY.staticcall(
                    abi.encode(
                        abi.encodePacked(txHash),
                        signatures[i],
                        signerPubKeys[i]
                    )
                );
            if (success && abi.decode(result, (bool))){
                validCount++;
            }
        }

        require(validCount >= threshold, "Insufficient PQC sigs");

        (bool ok, bytes memory ret) = target.call{value: value}(data);
        require(ok, "Execution failed");
        return ret;
    }
}
```

Developer Usage: PQC operations from a CosmWasm contract:

```
use cosmwasm_std::{DepsMut, Response, Binary};
use qorechain_pqc::{MlDsa87Verifier, Shake256Hasher};

/// Verify an ML-DSA-87 signature within a CosmWasm contract
pub fn verify_pqc_signature(
    deps: DepsMut,
    message: Binary,
    signature: Binary,
    public_key: Binary,
) -> Result<Response, ContractError> {
    // Call the native PQC verification module
    let is_valid = MlDsa87Verifier::verify(
        deps.api,
        &message,
        &signature,
        &public_key,
    )?;

    if !is_valid {
        return Err(ContractError::InvalidPqcSignature {});
    }

    // Compute a SHAKE-256 hash of the verified message
    let hash = Shake256Hasher::hash(deps.api, &message, 32)?;

    Ok(Response::new()
        .add_attribute("action", "pqc_verify")
        .add_attribute("algorithm", "ML-DSA-87")
        .add_attribute("message_hash", hex::encode(&hash))
        .add_attribute("verified", "true"))
}
```

4.4 Hybrid Classical-PQC Framework

4.4.1 Design Rationale

The hybrid classical-PQC framework serves four purposes, enabling pragmatic migration while maintaining security and institutional acceptance. Hybrid signatures provide defense-in-depth against both classical and quantum adversaries while maintaining backward compatibility with existing infrastructure. The framework allows institutions to transition gradually from classical-only systems (ECDSA) through hybrid systems to pure post-quantum systems (ML-DSA-87), without requiring ecosystem-wide coordination or creating security gaps during transition periods.

1. **Defense in depth**: If lattice assumptions are weakened by future mathematical or quantum advances (e.g., a cryptanalytic breakthrough that reduces lattice security from 128-bit to 80-bit), the classical signature component (ECDSA with

128-bit classical security) still provides security against classical adversaries. The scheme is secure against either a quantum or a very-well-funded classical adversary, but not both simultaneously.

2. **Backward compatibility**: Existing tools, wallets, and libraries that only support ECDSA can interact with QoreChain during the migration window (default 12 months, governance-adjustable). Users with ECDSA-only wallets can continue submitting transactions via the hybrid pathway, gaining immediate PQC protection without waiting for wallet updates. New wallets can be optimized for ML-DSA-87 alone, dropping ECDSA support after the migration window closes, reducing code complexity and improving performance.

3. **Regulatory and institutional credibility**: Some regulatory frameworks (especially in conservative industries like finance and healthcare) may require hybrid approaches until PQC-only schemes accumulate sufficient deployment history and institutional acceptance. The hybrid approach allows institutions to adopt QoreChain with assurance that security comes from both well-understood classical cryptography and post-quantum schemes, reducing perceived risk during the transition period.

4. **Ecosystem breadth**: The hybrid framework enables simultaneous deployment across multiple user types: early adopters using PQC-only keys, conservative institutions using hybrid keys, and legacy systems using classical-only keys (through special compatibility modules), without requiring ecosystem-wide coordination.

4.4.2 Hybrid Signature Construction

A hybrid signature concatenates both a classical and a PQC signature:

$$\sigma_{\text{hybrid}} = (\sigma_{\text{ECDSA}}, \sigma_{\text{ML-DSA-87}}) \tag{4.22}$$

Verification requires both components to validate:

$$\text{Verify}_{\text{hybrid}}(pk, m, \sigma) = \text{Verify}_{\text{ECDSA}}(pk_c, m, \sigma_c) \wedge \text{Verify}_{\text{ML-DSA}}(pk_q, m, \sigma_q) \tag{4.23}$$

The security of the hybrid scheme is:

$$\text{Adv}^{\text{EU-CMA}}_{\text{hybrid}}(\mathcal{A}) \leq \min\left(\text{Adv}^{\text{EU-CMA}}_{\text{ECDSA}}(\mathcal{A}),\ \text{Adv}^{\text{EU-CMA}}_{\text{ML-DSA-87}}(\mathcal{A})\right) \tag{4.24}$$

The hybrid scheme is at least as secure as the stronger of the two component schemes. An adversary must break both ECDLP (for ECDSA) and MSIS/MLWE (for ML-DSA-87) to forge a signature.

4.4.3 Migration Protocol

The migration from classical to PQC-only proceeds through three governance-controlled phases:

- **Phase 1** (Pre-launch / Legacy): Classical ECDSA signatures are accepted. This phase exists for backward compatibility with pre-genesis tooling.

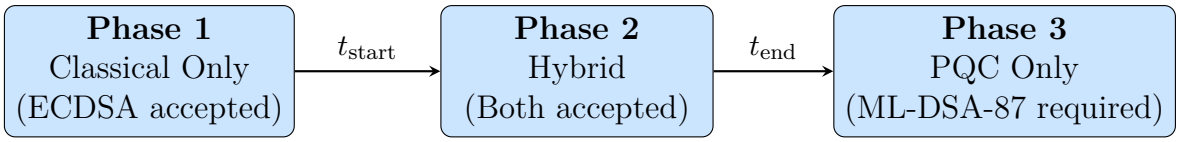

Figure 4.2: Three-phase cryptographic migration protocol

- **Phase 2** (Hybrid): Both classical ECDSA and ML-DSA-87 signatures are accepted. Hybrid dual-signatures are recommended. This is the launch configuration.
- **Phase 3** (PQC-Only): Only ML-DSA-87 signatures are accepted. Classical ECDSA is deprecated. Transition to Phase 3 requires a governance vote.

4.4.4 Asset Migration

Users holding assets under classical ECDSA keys can migrate to PQC-protected accounts through a signed migration transaction:

```
# Migrate an existing ECDSA wallet to ML-DSA-87
$ qorechaind tx pqc migrate-account \
    --from my-ecdsa-wallet \
    --new-algo ml-dsa-87 \
    --chain-id qorechain-diana

# The migration transaction:
# 1. Signs with the old ECDSA key (proving ownership)
# 2. Registers a new ML-DSA-87 public key
# 3. Transfers all assets to the new PQC-protected address
# 4. Marks the old address as migrated (no further classical txs)

Migration successful:
  Old address: qor1abc... (ECDSA)
  New address: qor1xyz... (ML-DSA-87)
  Assets transferred: 15,000,000 uqor
```

4.5 Cryptographic Agility Framework

4.5.1 Algorithm Registry

QoreChain maintains an on-chain registry of all supported cryptographic algorithms. Each entry specifies:

```
// On-chain algorithm registry query
$ qorechaind query pqc algorithms

algorithms:
- id: "ml-dsa-87"
  standard: "FIPS 204"
  category: "signature"
  security_level: 5
  status: "active"
  activated_at_height: 1
  parameters:
    public_key_size: 2592
    signature_size: 4595
    security_bits_classical: 256
    security_bits_quantum: 128

- id: "ml-kem-1024"
  standard: "FIPS 203"
  category: "kem"
  security_level: 5
  status: "active"
  activated_at_height: 1
  parameters:
    public_key_size: 1568
    ciphertext_size: 1568
    shared_secret_size: 32

- id: "slh-dsa-shake-256f"
  standard: "FIPS 205"
  category: "signature"
  security_level: 5
  status: "active"
  activated_at_height: 1
  parameters:
    public_key_size: 64
    signature_size: 49856
```

4.5.2 Hot-Swap Protocol

Algorithm transitions proceed without network downtime through an overlap window:

$$\text{AcceptedAlgorithms}(h) = \begin{cases} \{A_{\text{old}}\} & \text{if } h < h_{\text{start}} \\ \{A_{\text{old}}, A_{\text{new}}\} & \text{if } h_{\text{start}} \leq h \leq h_{\text{end}} \\ \{A_{\text{new}}\} & \text{if } h > h_{\text{end}} \end{cases} \tag{4.25}$$

The transition parameters $(A_{\text{new}}, h_{\text{start}}, h_{\text{end}})$ are set by governance vote. The over-

lap window must be long enough for all active validators and users to rotate their keys:

$$h_{\text{end}} - h_{\text{start}} \geq \max\left(T_{\text{validator\_rotation}}, T_{\text{user\_migration}}ation\right) \tag{4.26}$$

4.5.3 Future-Proofing

The cryptographic agility framework accommodates future NIST selections and ongoing cryptanalytic advances over 50+ year timescales. Rather than committing to a single post-quantum algorithm family for eternity, the governance framework allows the network to activate alternative algorithms as new candidates reach standardization or as existing assumptions weaken. Code-based, lattice-based, multivariate-based, and hash-based algorithms each rest on different mathematical foundations; advances in one family (e.g., lattice cryptanalysis) leave others unaffected. QoreChain maintains compatibility for multiple algorithm families, enabling migration without consensus disruption.

- **HQC** (code-based KEM): Under NIST review as an additional KEM candidate, based on quasi-cyclic codes rather than lattices, providing assumption diversity.
- **BIKE** (code-based KEM): Based on quasi-cyclic moderate-density parity-check codes.
- **Classic McEliece**: A code-based KEM with a long track record but large key sizes (261 KB public key), suitable for long-term key establishment where key size is not a constraint.

New algorithms are proposed through governance, undergo community review and cryptanalytic scrutiny, and are activated through the standard hot-swap protocol. This ensures no algorithm is forced upon the network before sufficient vetting.

4.6 Performance Analysis

4.6.1 Computational Overhead

PQC algorithms introduce measurable but manageable overhead compared to classical alternatives:

Algorithm	PK Size	Sig/CT Size	Sign/Encap	Verify/Decap	Security
ECDSA (secp256k1)	33 B	64 B	0.05 ms	0.07 ms	Classical only
ML-DSA-87	2,592 B	4,595 B	1.5 ms	0.5 ms	NIST Cat. 5
SLH-DSA-256f	64 B	49,856 B	3 ms	2 ms	NIST Cat. 5
ECDH (X25519)	32 B	32 B	0.05 ms	0.05 ms	Classical only
ML-KEM-1024	1,568 B	1,568 B	0.3 ms	0.4 ms	NIST Cat. 5

Table 4.5: Comparative performance: classical vs. PQC algorithms

The primary cost is signature size: ML-DSA-87 signatures are approximately 72× larger than ECDSA. For a block containing m transactions, the additional bandwidth is:

$$\Delta B_{\text{block}} = m \cdot (|\sigma_{\text{ML-DSA-87}}| - |\sigma_{\text{ECDSA}}|) = m \cdot 4{,}531 \text{ bytes} \tag{4.27}$$

For 5,000 transactions per block, this is approximately 22.7 MB of additional signature data, significant but within the capacity of modern network connections.

4.6.2 QCAI-Driven Optimization

QCAI applies several techniques to reduce the effective overhead of PQC operations:

Batched Verification. ML-DSA-87 supports batch verification where multiple signatures are verified more efficiently than individual verification:

$$T_{\text{batch}}(m) = T_{\text{setup}} + m \cdot T_{\text{marginal}} < m \cdot T_{\text{individual}} \tag{4.28}$$

The setup cost is amortized across the batch, yielding approximately 30% speedup for batches of 100+ signatures.

Parallel Processing. Signature verification is embarrassingly parallel. QCAI schedules verification tasks across available cores to maintain block processing within the target time window.

Compression for Light Nodes. When synchronizing state to light nodes, QCAI applies signature aggregation and proof compression to reduce bandwidth requirements by up to 60%.

4.6.3 Storage Projections

Larger PQC signatures increase long-term storage requirements. Assuming an average of 3,000 transactions per block and 2-second block times:

$$\text{Annual storage}_{\text{sigs}} = 3{,}000 \times 4{,}595 \times \frac{365 \times 24 \times 3{,}600}{2} \approx 217 \text{ TB/year (theoretical max)} \tag{4.29}$$

State pruning, checkpoint-based synchronization, and signature aggregation reduce the effective storage requirement for full nodes to a fraction of this theoretical maximum. Light nodes store only block headers and proofs, requiring orders of magnitude less storage.

Pruning strategies: Full nodes implement three complementary storage optimization techniques. Signature aggregation reduces per-signature size by combining multiple signatures over the same data using batch verification; for a typical block with 100 transactions, aggregation reduces signature storage overhead by 40 to 50 percent. Checkpoint-based pruning allows nodes to discard transaction history older than the last finality checkpoint (roughly 1 hour at 2-second block times) while retaining state proofs sufficient to verify all current account balances; this reduces full node storage to approximately 200 GB per year rather than 217 TB. Merkle tree path compression stores only tree roots and necessary paths to leaf values rather than complete trees, further reducing overhead by an additional 30 percent.

Storage growth analysis: Storage requirements scale predictably with network throughput. At 5,000+ TPS (design target), annual storage reaches approximately 722 TB for raw signatures without optimization. With the three techniques above applied, effective full-node storage drops to 150 to 200 GB per year, a compression

ratio of 3,600:1. Archive nodes (required for historical auditing and light node verification) retain full history and require petabyte-scale storage within 5 to 10 years of network operation, representing an acceptable cost for large exchanges, custodians, and blockchain research institutions.

Light node requirements: Light nodes implement a radically different storage model: they store only block headers (approximately 1 KB per block) and a minimal set of proofs required to verify their own account state. At 2-second blocks, light node storage reaches only 15 to 20 MB per year, making quantum-safe blockchain accessible on resource-constrained devices (IoT, mobile, embedded systems) without full blockchain history.

4.7 Formal Security Analysis

4.7.1 Security Model

QoreChain's PQC integration is analyzed under the following formal security model:

Threat model. The adversary $\mathcal{A}$ is a quantum polynomial-time (QPT) algorithm with:

- Access to a quantum computer with at most 2^{128} quantum gates
- Adaptive access to signing and decapsulation oracles
- The ability to corrupt up to $f < n/3$ validators
- Full knowledge of the protocol specification and all public keys

Security goals:

1. **Transaction integrity**: No QPT adversary can forge a valid transaction signature
2. **Channel confidentiality**: No QPT adversary can recover the plaintext of encrypted communications
3. **State authenticity**: No QPT adversary can produce a valid Merkle proof for a false state value
4. **Consensus safety**: No QPT adversary controlling fewer than $n/3$ validators can cause a safety violation

Theorem 1 (Transaction Security). Under the MLWE and MSIS assumptions with parameters as specified for ML-DSA-87, for any QPT adversary $\mathcal{A}$ making at most q_S signing queries and q_H hash queries:

$$\mathrm{Adv}^{\mathrm{forge}}_{\mathrm{QoreChain}}(\mathcal{A}) \leq \mathrm{Adv}^{\mathrm{MSIS}}(\mathcal{B}) + \mathrm{Adv}^{\mathrm{MLWE}}(\mathcal{B}') + \frac{q_H + q_S}{2^{256}} \leq \mathrm{negl}(\lambda) \tag{4.30}$$

Theorem 2 (Cross-Chain Security). For any QPT adversary $\mathcal{A}$ attempting to forge or read a cross-chain packet:

$$\text{Adv}^{\text{forge}} + read_{\text{QCB}}(\mathcal{A}) \leq \text{Adv}^{\text{IND-CCA2}}_{\text{ML-KEM}}(\mathcal{B}) + \text{Adv}^{\text{EU-CMA}}_{\text{ML-DSA}}(\mathcal{B}') \leq \text{negl}(\lambda) \tag{4.31}$$

Theorem 3 (State Integrity). For a Merkle tree of depth d with SHAKE-256 internal nodes, the probability that a QPT adversary produces a valid proof for a false leaf value is:

$$\Pr[\text{FalseProof}] \leq d \cdot \text{Adv}^{\text{coll}}_{\text{SHAKE-256}}(\mathcal{A}) \leq d \cdot 2^{-128} \tag{4.32}$$

For $d = 30$ (supporting $> 10^9$ state entries), this probability is bounded by $30 \cdot 2^{-128} \approx 2^{-123}$, negligible by any practical measure.

4.7.2 Known Attack Vectors and Mitigations

Attack Vector	**Risk**	**Mitigation**
Shor's algorithm on ECDSA	Signature forgery	ML-DSA-87 as primary scheme
Side-channel on PQC	Key extraction	Constant-time code, HSM isolation
Lattice reduction advances	Reduced margin	SLH-DSA fallback (hash-based)
Implementation bugs	Incorrect verification	Formal verification of critical paths
Harvest Now, Decrypt Later	Historical exposure	ML-KEM-1024 for all channels from genesis
Algorithm obsolescence	Future standards	Cryptographic agility, hot-swap protocol

Table 4.6: PQC attack vectors and QoreChain mitigations

Chapter 5

QCAI: AI-Native Intelligence Layer

5.1 Architecture Overview

QoreChain integrates artificial intelligence as a first-class network primitive rather than an external service. The QCAI layer provides deterministic, on-chain AI inference for network operations alongside off-chain model serving for developer-facing applications. This dual-mode architecture separates consensus-critical AI (which must be deterministic and verifiable) from advisory AI (which tolerates non-determinism in exchange for richer capabilities).

5.1.1 Design Principles

The QCAI architecture is governed by three core principles:

1. **Deterministic On-Chain Inference:** All AI operations that affect consensus state use quantized, fixed-point models to guarantee identical outputs across all validators.

2. **Tiered Service Model:** Three service tiers (Fast, Balanced, Advanced) allow developers to select the appropriate cost-latency-quality tradeoff for each use case.

3. **Verifiable Computation:** Every AI inference produces a cryptographic commitment that can be verified without re-executing the model.

Tier	Designation	Latency Target	Context Window	Use Cases
QCAI Fast	Real-time inference	< 200 ms	32K tokens	Transaction classification, threat detection
QCAI Balanced	Standard inference	< 2 s	128K tokens	Contract generation, code auditing
QCAI Advanced	Deep analysis	< 30 s	256K tokens	Formal verification, complex optimization

Table 5.1: QCAI service tier specifications

5.1.2 Processing Architecture

The separation between on-chain and off-chain AI processing is formalized as follows. Let $\mathcal{M}_{\text{on}}$ denote the set of on-chain models and $\mathcal{M}_{\text{off}}$ the off-chain models. For any inference request r, the routing function ρ is defined as:

$$\rho(r) = \begin{cases} \mathcal{M}_{\text{on}} & \text{if } r \in \mathcal{C}_{\text{consensus}} \text{ (consensus-critical)} \\ \mathcal{M}_{\text{off}} & \text{if } r \in \mathcal{C}_{\text{advisory}} \text{ (advisory)} \end{cases} \tag{5.1}$$

where $\mathcal{C}_{\text{consensus}}$ includes transaction routing, fee estimation, threat detection, and validator selection, while $\mathcal{C}_{\text{advisory}}$ includes contract generation, security auditing, and developer assistance.

For on-chain models, determinism is enforced through fixed-point quantization. Given a floating-point model $f : \mathbb{R}^n \to \mathbb{R}^m$, the quantized model $\hat{f} : \mathbb{Z}^n \to \mathbb{Z}^m$ is constructed as:

$$\hat{f}(x) = \text{round}\left(\frac{f(s \cdot x)}{s}\right), \quad s = 2^{16} \tag{5.2}$$

where s is the scaling factor that maps between fixed-point and floating-point representations. This guarantees bit-identical outputs across heterogeneous hardware.

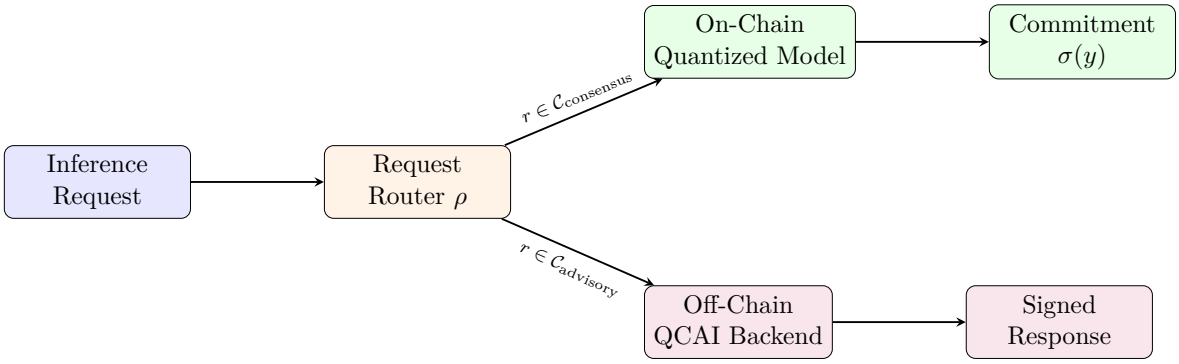

Figure 5.1: QCAI dual-mode processing architecture

Querying QCAI Service Tiers (CLI)

```
# List available QCAI tiers and current pricing
qorchaind query qcai tiers

# Check QCAI service status and model versions
qorchaind query qcai status --tier balanced

# Query your QCAI usage and remaining credits
qorchaind query qcai usage --from qor1abc...def
```

5.2 AI-Driven Network Operations

5.2.1 Intelligent Transaction Routing

The QCAI transaction router optimizes block packing, cross-VM routing, and multi-layer assignment (main chain, sidechain, paychain, or rollup) using a reinforcement

learning agent trained on historical transaction patterns. For rollup-bound transactions, the router additionally selects the optimal rollup instance based on the transaction's profile affinity and current rollup congestion. The routing policy π^* is learned via Q-learning with the following Bellman optimality equation:

$$Q^*(s,a) = \mathbb{E}\left[R(s,a) + \gamma \max_{a'} Q^*(s',a')\right] \tag{5.3}$$

where the state s encodes the current mempool composition, pending cross-VM calls, network congestion metrics, and active rollup instance utilization. The action space $\mathcal{A}$ represents possible transaction orderings, VM assignments, and layer routing decisions (including rollup instance selection for RDK-managed rollups). The reward function R balances throughput, latency, and fee revenue:

$$R(s,a) = \alpha \cdot \text{TPS}(s,a) + \beta \cdot \frac{1}{\text{Latency}(s,a)} + \gamma_f \cdot \text{FeeRevenue}(s,a) - \lambda \cdot \text{RevertRate}(s,a) \tag{5.4}$$

with hyperparameters $\alpha, \beta, \gamma_f, \lambda > 0$ governing the tradeoff between throughput maximization, latency minimization, fee optimization, and revert penalty.

The Q-function is approximated by a deep neural network Q_θ with parameters θ, trained to minimize the temporal difference loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(r+\gamma \max_{a'} Q_{\theta^-}(s',a') - Q_\theta(s,a)\right)^2\right] \tag{5.5}$$

where $\mathcal{D}$ is the experience replay buffer and θ^- are the target network parameters updated via Polyak averaging:

$$\theta^- \leftarrow \tau\theta + (1-\tau)\theta^- \tag{5.6}$$

5.2.2 Dynamic Validator Selection

QCAI enhances the validator selection process by scoring validators on a multidimensional performance vector. For validator v_i, the composite score $S(v_i)$ is computed as:

$$S(v_i) = \sum_{k=1}^{K} w_k \cdot \phi_k(v_i) \tag{5.7}$$

where $\phi_k(v_i)$ are normalized feature functions measuring uptime, latency, stake weight, historical slash rate, geographic diversity, and PQC compliance. The weights w_k are learned via gradient descent on historical block production quality:

$$w_k^{(t+1)} = w_k^{(t)} - \eta \cdot \frac{\partial \mathcal{L}_{\text{block}}}{\partial w_k} \tag{5.8}$$

The block quality loss $\mathcal{L}_{\text{block}}$ penalizes missed blocks, late proposals, and censored transactions:

$$\mathcal{L}_{\text{block}} = \sum_{b \in \mathcal{B}} (\mathbb{1}[\text{missed}(b)] + \alpha_t \cdot \text{delay}(b) + \beta_c \cdot \text{censorship\_score}(b)) \tag{5.9}$$

Querying AI-Enhanced Validator Scores (SDK)

```
import { QoreChainClient } from "@qorechain/sdk";

const client = new QoreChainClient({ rpcEndpoint:
    ↪ "https://rpc.qorechain.io" });

// Get AI-scored validator rankings
const rankings = await client.qcai.getValidatorRankings({
  sortBy: "composite_score",
  includeMetrics: true,
  limit: 20,
});

for (const v of rankings.validators) {
  console.log(
    `${v.moniker}: score=${v.compositeScore.toFixed(4)}, ` +
    `uptime=${v.uptime}%, latency=${v.avgLatencyMs}ms, ` +
    `pqc=${v.pqcCompliant ? "yes" : "no"}`
  );
}

// Get routing optimization suggestions for a transaction
const routingAdvice = await client.qcai.getRoutingAdvice({
  msgType: "/qorechain.evm.v1.MsgEthereumTx",
  gasEstimate: 250000,
  priority: "high",
});
console.log(`Recommended VM: ${routingAdvice.targetVM}`);
console.log(`Recommended layer: ${routingAdvice.targetLayer}`); //
    ↪ main/side/pay/rollup
console.log(`Estimated inclusion: block
    ↪ +${routingAdvice.blocksToInclusion}`);

// Get AI-optimized rollup configuration suggestion
const rollupSuggestion = await client.qcai.suggestRollupProfile({
  workloadType: "defi",
  expectedTPS: 500,
  avgTxSize: 256,
});
console.log(`Suggested profile: ${rollupSuggestion.profile}`);
console.log(`Settlement mode: ${rollupSuggestion.settlementMode}`);
console.log(`DA backend: ${rollupSuggestion.daBackend}`);
```

5.2.3 Predictive Resource Allocation

Network resource allocation leverages Long Short-Term Memory (LSTM) networks to forecast demand across the three virtual machines. Given a time series of resource utilization $\{x_t\}_{t=1}^{T}$, the LSTM computes hidden states h_t via the following gating mechanism:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad \text{(forget gate)} \tag{5.10}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad \text{(input gate)} \tag{5.11}$$
$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad \text{(candidate cell)} \tag{5.12}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad \text{(cell state)} \tag{5.13}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad \text{(output gate)} \tag{5.14}$$
$$h_t = o_t \odot \tanh(c_t) \quad \text{(hidden state)} \tag{5.15}$$

where σ is the sigmoid function, $\odot$ denotes element-wise multiplication, and W_f, W_i, W_c, W_o are learned weight matrices. The forecast for τ steps ahead is:

$$\hat{x}_{t+\tau} = W_{\text{proj}} \cdot h_t + b_{\text{proj}} \tag{5.16}$$

The LSTM is trained to minimize the mean squared forecast error with an L2 regularization penalty:

$$\mathcal{L}_{\text{LSTM}} = \frac{1}{T} \sum_{t=1}^{T} \|x_{t+\tau} - \hat{x}_{t+\tau}\|^2 + \lambda_r \sum_{W \in \Theta} \|W\|_F^2 \tag{5.17}$$

where $\|\cdot\|_F$ is the Frobenius norm and Θ is the full parameter set.

5.2.4 Predictive Fee Optimization

QCAI computes dynamic fee recommendations by combining the LSTM demand forecast with an economic model of fee elasticity. The recommended base fee $\hat{B}_{t+1}$ for the next block is:

$$\hat{B}_{t+1} = B_t \cdot \left(1 + \frac{1}{d} \cdot \frac{G_t - G^*}{G^*}\right) \tag{5.18}$$

where B_t is the current base fee, G_t is the actual gas used in the current block, G^* is the target gas utilization, and d is the fee adjustment denominator controlling smoothness. The AI enhancement adds a predictive correction term Δ_{AI}:

$$\hat{B}_{t+1}^{\text{AI}} = \hat{B}_{t+1} + \Delta_{\text{AI}}, \quad \Delta_{\text{AI}} = \eta_f \cdot (\hat{G}_{t+1} - G^*) \tag{5.19}$$

where $\hat{G}_{t+1}$ is the LSTM-predicted gas demand for the next block.

AI-Enhanced Fee Estimation (SDK)

```
import { QoreChainClient } from "@qorechain/sdk";

const client = new QoreChainClient({ rpcEndpoint:
    ↪ "https://rpc.qorechain.io" });
```

```
// Get AI-predicted fee estimates for the next N blocks
const feeEstimate = await client.qcai.estimateFees({
  horizonBlocks: 5,
  confidenceLevel: 0.95,
});

console.log(`Current base fee: ${feeEstimate.currentBaseFee} uqor`);
console.log(`Predicted (next block):
    ↪ ${feeEstimate.predictions[0].fee} uqor`);
console.log(`95% CI: [${feeEstimate.predictions[0].ciLow},
    ↪ ${feeEstimate.predictions[0].ciHigh}]`);

// Get per-VM fee breakdown
for (const vm of feeEstimate.vmBreakdown){
  console.log(` ${vm.name}: congestion=${vm.congestionIndex},
      ↪ fee=${vm.suggestedFee} uqor`);
}
```

5.2.5 AI-Enhanced Threat Detection

The QCAI threat detection system employs a two-stage architecture: an autoencoder for anomaly detection followed by a Graph Neural Network (GNN) for transaction graph analysis.

Stage 1: Autoencoder Anomaly Detection. The autoencoder learns a compressed representation of normal transaction patterns. For an input feature vector $x \in \mathbb{R}^n$, the encoder E_ϕ and decoder D_ψ are defined as:

$$z = E_\phi(x) = \sigma(W_e x + b_e), \quad z \in \mathbb{R}^d,\ d \ll n \tag{5.20}$$

$$\hat{x} = D_\psi(z) = \sigma(W_d z + b_d), \quad \hat{x} \in \mathbb{R}^n \tag{5.21}$$

The reconstruction loss for a single transaction is:

$$\mathcal{L}_{\text{recon}}(x) = \|x - \hat{x}\|^2 = \|x - D_\psi(E_\phi(x))\|^2 \tag{5.22}$$

A transaction is flagged as anomalous if its reconstruction error exceeds the threshold τ_{anom}:

$$\text{Anomaly}(x) = \mathbb{1}\left[\mathcal{L}_{\text{recon}}(x) > \tau_{\text{anom}}\right], \quad \tau_{\text{anom}} = \mu_{\mathcal{L}} + k \cdot \sigma_{\mathcal{L}} \tag{5.23}$$

where $\mu_{\mathcal{L}}$ and $\sigma_{\mathcal{L}}$ are the mean and standard deviation of reconstruction losses over the training set, and k controls the sensitivity (typically $k = 3$ for 99.7% confidence under Gaussian assumptions).

Stage 2: Graph Neural Network Analysis. Flagged transactions are analyzed in the context of the transaction graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where vertices represent addresses and edges represent fund flows. The GNN performs message passing to aggregate neighborhood information:

$$h_v^{(\ell+1)} = \text{UPDATE}\left(h_v^{(\ell)}, \text{AGG}\left(\{m_{u\to v}^{(\ell)} : u \in \mathcal{N}(v)\}\right)\right) \tag{5.24}$$

where the message from neighbor u to node v at layer ℓ is:

$$m_{u\to v}^{(\ell)} = \text{MSG}^{(\ell)}(h_u^{(\ell)}, h_v^{(\ell)}, e_{u,v}) \tag{5.25}$$

with $e_{u,v}$ encoding edge features (transfer amount, frequency, token type). After L layers of message passing, the final node embedding $h_v^{(L)}$ is used for threat classification:

$$P(\text{threat} \mid v) = \text{softmax}(W_{\text{cls}} \cdot h_v^{(L)} + b_{\text{cls}}) \tag{5.26}$$

The threat categories include: wash trading, Sybil attack patterns, flash loan exploits, sandwich attacks, and governance manipulation.

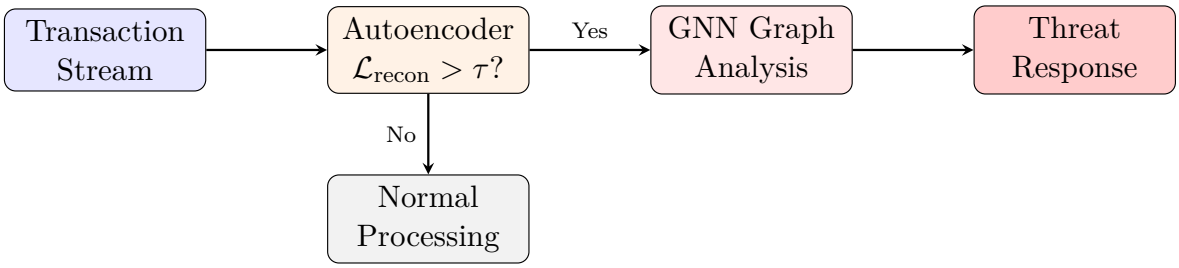

Figure 5.2: Two-stage AI threat detection pipeline

Querying Threat Detection Status (CLI)

```
# Check threat detection status for an address
qorchaind query qcai threat-score qor1abc...def

# Get network-wide threat summary
qorchaind query qcai threat-summary --last-blocks 100

# Subscribe to real-time threat alerts (WebSocket)
qorchaind query qcai subscribe-threats --min-severity medium \
  --output json
```

5.3 QoreChain Studio

QoreChain Studio is an AI-powered development environment that enables natural-language smart contract generation and automated security auditing across 17 supported blockchains: QoreChain, Cosmos, Ethereum, Solana, Avalanche, Arbitrum, Optimism, Polygon, BSC, Cardano, Polkadot, NEAR, TON, Tezos, Sui, Aptos, and TRON.

5.3.1 Natural-Language Contract Generation

The contract generation pipeline transforms a natural-language specification $\mathcal{S}$ into a verified smart contract $\mathcal{C}$ through a multi-stage process:

$$\mathcal{S} \xrightarrow{\text{Parse}} \mathcal{A}_{\text{AST}} \xrightarrow{\text{Generate}} \mathcal{C}_{\text{draft}} \xrightarrow{\text{Audit}} \mathcal{C}_{\text{verified}} \xrightarrow{\text{Optimize}} \mathcal{C}_{\text{final}} \tag{5.27}$$

Each stage applies a specialized QCAI model:

1. **Semantic Parsing:** Extracts structured intent from the natural-language specification using a fine-tuned transformer that maps $\mathcal{S}$ to an abstract syntax tree $\mathcal{A}_{\text{AST}}$.
2. **Code Generation:** Synthesizes target-language code from the AST, applying chain-specific patterns and gas optimization heuristics.
3. **Security Auditing:** Scans the generated code against a vulnerability database of 200+ known attack patterns.
4. **Gas/Resource Optimization:** Applies chain-specific optimizations (EVM gas, CosmWasm CPU cycles, SVM compute units).

5.3.2 Vulnerability Scoring Model

The security auditing stage assigns a vulnerability score to each detected issue. For a potential vulnerability v with features $\mathbf{f}_v$, the severity score is:

$$\text{Severity}(v) = \sigma\left(\mathbf{w}^T \mathbf{f}_v + b\right) \cdot \text{Impact}(v) \cdot \text{Exploitability}(v) \tag{5.28}$$

where $\text{Impact}(v) \in [0, 1]$ estimates the potential financial loss (normalized by contract value) and $\text{Exploitability}(v) \in [0, 1]$ estimates the likelihood of successful exploitation. The feature vector $\mathbf{f}_v$ encodes:

$$\mathbf{f}_v = [\text{pattern\_match}, \text{data\_flow\_depth}, \text{control\_flow\_complexity}, \text{external\_calls}, \text{reentrancy\_risk}, \text{overflow\_risk}]^T \tag{5.29}$$

The overall contract security score $\mathcal{Q}(\mathcal{C})$ aggregates individual vulnerability scores:

$$\mathcal{Q}(\mathcal{C}) = 1 - \max_{v \in \mathcal{V}(\mathcal{C})} \text{Severity}(v) - \lambda_a \cdot \frac{|\mathcal{V}(\mathcal{C})|}{|\mathcal{L}(\mathcal{C})|} \tag{5.30}$$

where $\mathcal{V}(\mathcal{C})$ is the set of detected vulnerabilities, $|\mathcal{L}(\mathcal{C})|$ is the number of lines, and λ_a is a penalty coefficient for vulnerability density.

5.3.3 Supported Contract Types

QoreChain Studio supports the following contract categories across all 17 chains:

Category	Contract Types	Languages	Audit Coverage
DeFi	AMM, lending, yield vault, stablecoin	Solidity, Rust, Move	45 vulnerability patterns
Token	ERC-20, ERC-721, ERC-1155, CW-20, SPL	Solidity, Rust	28 vulnerability patterns
Governance	DAO, multisig, timelock, voting	Solidity, Rust	22 vulnerability patterns
Bridge	Lock-mint, burn-bridge, atomic swap	Solidity, Rust	35 vulnerability patterns
Identity	DID, credential, access control	Solidity, Rust	18 vulnerability patterns

Table 5.2: QoreChain Studio contract types and audit coverage

QoreChain Studio: Contract Generation (API)

```
import { QoreChainStudio } from "@qorechain/studio-sdk";

const studio = new QoreChainStudio({
  apiKey: process.env.QORECHAIN_API_KEY,
  tier: "balanced", // "fast" | "balanced" | "advanced"
});

// Generate a contract from natural language
const result = await studio.generateContract({
  description: "ERC-20 token with 1 billion supply, 2% transfer tax
    ↪ " +
              "sent to a treasury address, pausable by owner, and " +
              "anti-whale max holding of 1% total supply",
  targetChain: "ethereum",
  language: "solidity",
  optimizeGas: true,
});

console.log(`Contract generated: ${result.contractName}`);
console.log(`Security score: ${result.securityScore}/1.00`);
console.log(`Gas estimate (deploy): ${result.gasEstimate.deploy}`);
console.log(`Vulnerabilities found:
    ↪ ${result.vulnerabilities.length}`);

// Print the generated Solidity code
console.log(result.sourceCode);
```

QoreChain Studio: Security Audit (CLI)

```
# Audit an existing contract file
qorechain-studio audit ./contracts/MyToken.sol \
  --chain ethereum \
  --tier advanced \
  --output report.json

# Audit a deployed contract by address
qorechain-studio audit-deployed 0xAbC...123 \
  --chain polygon \
```

```
  --include-bytecode-analysis

# Batch audit all contracts in a directory
qorechain-studio audit ./contracts/ \
  --recursive \
  --format markdown \
  --output audit-report.md
```

QoreChain Studio: Multi-Chain Generation (Rust SDK)

```rust
use qorechain_studio::{StudioClient, GenerateRequest, TargetChain};

#[tokio::main]
async fn main() -> Result<(), Box<dyn std::error::Error>> {
    let studio = StudioClient::new(
        std::env::var("QORECHAIN_API_KEY")?,
    );

    // Generate the same contract logic for multiple chains
    let chains = vec![
        TargetChain::Ethereum,
        TargetChain::QoreChainCosmWasm,
        TargetChain::Solana,
    ];

    let spec = "Escrow contract: buyer deposits funds, seller
        ↪ delivers, \
                arbiter resolves disputes. 1% platform fee. \
                30-day auto-refund if unresolved.";

    for chain in &chains {
        let result = studio.generate(GenerateRequest {
            description: spec.to_string(),
            target_chain: chain.clone(),
            optimize: true,
            audit: true,
        }).await?;

        println!("[{:?}] Score: {:.2}, Vulns: {}",
            chain, result.security_score,
                ↪ result.vulnerabilities.len());
    }

    Ok(())
}
```

5.4 Mathematical Foundations of QCAI Models

This section formalizes the core machine learning architectures employed by QCAI.

5.4.1 Transformer Attention with Security Context

The QCAI contract generation models use a modified transformer attention mechanism that incorporates a security context matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, where S_{ij} encodes the known security relationship between token positions i and j (e.g., whether position j is in a security-critical control flow path relative to i).

The standard scaled dot-product attention is:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{5.31}$$

The security-aware variant modifies this as:

$$\text{SecAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{S}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} + \alpha_s \cdot \mathbf{S}\right)\mathbf{V} \tag{5.32}$$

where α_s is a learnable scaling parameter that controls the influence of security context on attention weights. Multi-head attention extends this with H parallel heads:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{S}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_H)\mathbf{W}^O \tag{5.33}$$

$$\text{head}_i = \text{SecAttention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V, \mathbf{S}) \tag{5.34}$$

5.4.2 GNN Message Passing for Transaction Graphs

The GNN used in threat detection implements a variant of the Graph Attention Network (GAT). The attention coefficient between nodes u and v at layer ℓ is:

$$\alpha_{uv}^{(\ell)} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}^{(\ell)}h_u^{(\ell)} \| \mathbf{W}^{(\ell)}h_v^{(\ell)} \| e_{uv}]\right)\right)}{\sum_{w \in \mathcal{N}(v)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}^{(\ell)}h_w^{(\ell)} \| \mathbf{W}^{(\ell)}h_v^{(\ell)} \| e_{wv}]\right)\right)} \tag{5.35}$$

where $\|$ denotes concatenation, $\mathbf{a}$ is a learnable attention vector, and $\mathbf{W}^{(\ell)}$ is the layer-specific weight matrix. The node update rule is:

$$h_v^{(\ell+1)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv}^{(\ell)} \cdot \mathbf{W}^{(\ell)}h_u^{(\ell)}\right) \tag{5.36}$$

Multi-head graph attention aggregates H independent attention heads:

$$h_v^{(\ell+1)} = \Big\|_{h=1}^{H} \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{uv}^{(\ell,h)} \cdot \mathbf{W}^{(\ell,h)}h_u^{(\ell)}\right) \tag{5.37}$$

5.4.3 Inference Commitment Scheme

To enable verifiable AI inference, QCAI produces a cryptographic commitment for each inference result. Given model parameters θ, input x, and output $y = f_\theta(x)$, the commitment is:

$$C(y, \theta, x) = H_{\text{SHAKE-256}}\left(y \| H(\theta) \| H(x) \| \text{nonce}\right) \tag{5.38}$$

where H is SHAKE-256 (the network's standard hash function) and nonce is a random value published alongside the commitment. Verification requires re-computing:

$$\text{Verify}(C, y, \theta, x, \text{nonce}) = \left[C \stackrel{?}{=} H_{\text{SHAKE-256}}\left(y \| H(\theta) \| H(x) \| \text{nonce}\right)\right] \tag{5.39}$$

For on-chain models, the model hash $H(\theta)$ is stored in the QCAI module's state, enabling any validator to verify that the correct model version was used.

5.5 QCAI Token Economics

All QCAI services are paid in QOR tokens. The pricing model distinguishes between standard and premium tiers:

Access Level	Tiers Included	Payment Model
Standard	QCAI Fast, QCAI Balanced	Per-inference QOR fee
Premium	QCAI Advanced	Monthly QOR subscription + per-inference

Table 5.3: QCAI access levels and payment models

The per-inference fee $F(r)$ for a request r is computed as:

$$F(r) = F_{\text{base}}(\text{tier}) + F_{\text{compute}}(r) + F_{\text{data}}(r) \tag{5.40}$$

where F_{base} is a fixed tier-dependent base fee, $F_{\text{compute}}(r)$ scales with the computational resources consumed (measured in inference units), and $F_{\text{data}}(r)$ scales with input/output data size.

QCAI fee revenue follows the standard network fee distribution: 37% to validators, 30% burned, 20% to treasury, 10% to stakers, and 3% to light nodes.

Paying for QCAI Inference (SDK)

```
import { QoreChainClient } from "@qorechain/sdk";

const client = new QoreChainClient({
  rpcEndpoint: "https://rpc.qorechain.io",
  signer: wallet,
});

// Estimate QCAI inference cost before execution
const estimate = await client.qcai.estimateCost({
```

```
  tier: "balanced",
  operation: "contract_audit",
  inputSizeBytes: 15000,
});
console.log(`Estimated cost: ${estimate.totalFee} uqor`);

// Execute a paid QCAI inference
const result = await client.qcai.invoke({
  tier: "balanced",
  operation: "contract_audit",
  input: contractSource,
  maxFee: "5000000uqor", // fee cap for safety
});
console.log(`Actual cost: ${result.fee} uqor`);
console.log(`Tx hash: ${result.txHash}`);
```

5.6 Model Governance

QCAI model updates are governed through the on-chain governance module. All models are registered in a versioned on-chain registry, and updates require governance approval to prevent unauthorized model changes that could affect network behavior.

5.6.1 Model Registry

The model registry maintains a mapping from model identifiers to their metadata:

$$\mathcal{R} : \text{ModelID} \to (H(\theta), \text{version}, \text{accuracy}, \text{governance\_proposal\_id}) \tag{5.41}$$

Each registry entry records the SHAKE-256 hash of the model parameters, the semantic version, the validated accuracy on a canonical test set, and the governance proposal that authorized the model.

Model versioning and lineage: The registry maintains a complete version history for each model, allowing rollback to any previously approved version within one governance proposal cycle. Each version is cryptographically linked to its predecessor through inclusion of the parent version hash in the registry, creating an immutable audit trail. This versioning model supports rapid iteration (new versions can be proposed and approved within 24 hours) while maintaining governance accountability and enabling emergency rollback if a deployed model exhibits unexpected behavior in production.

On-chain registration process: Developers submit model registration proposals through the standard governance channel, specifying the model hash, version, benchmark results on a canonical test set (e.g., 10,000 representative transactions), and target deployment block height. Validators independently verify the model during the voting period by running it against the canonical test set and comparing claimed accuracy against observed results. The model must achieve minimum accuracy threshold (currently 95 percent) and must not degrade any security metric (false positive rate,

latency, resource consumption). Once approved, the model becomes immutable in the registry; subsequent updates require new governance proposals.

Access control and querying: All models in the registry are public and accessible through query interfaces; QCAI internally selects the appropriate model version based on block height and active governance decisions. External services can query the registry to verify which model version was active during any historical period, enabling external audits, regulatory compliance, and cross-chain model synchronization. The registry also tracks model dependencies: if one model version depends on embeddings or features from another model, the registry records this relationship, preventing orphaned models from being deleted and ensuring reproducibility of historical results.

5.6.2 Update Protocol

Model updates follow a three-phase protocol:

1. **Proposal:** A governance proposal specifies the new model hash $H(\theta')$, the target model ID, and benchmark results comparing the new model against the current production model.

2. **Validation Period:** During the voting period, validators independently verify the model on the canonical test set. A model must achieve accuracy $\geq \tau_{\text{acc}}$ (currently 95%) and must not degrade any safety metric.

3. **Activation:** Upon governance approval, the model is activated at a specified future block height, giving all validators time to download and cache the new model.

The rollback mechanism allows emergency reversion to any previously approved model version via an expedited governance vote with a shortened voting period.

Querying QCAI Model Registry (CLI)

```
# List all registered QCAI models
qorchaind query qcai models

# Get details for a specific model
qorchaind query qcai model tx-router --version 2.1.0

# Check pending model update proposals
qorchaind query gov proposals --status voting_period \
  --msg-type /qorechain.qcai.v1.MsgUpdateModel

# Submit a model update proposal (requires governance deposit)
qorchaind tx gov submit-proposal update-qcai-model \
  --model-id tx-router \
  --new-version 2.2.0 \
  --model-hash "sha3_256:abc123..." \
  --benchmark-report ./benchmark_v2.2.0.json \
  --deposit 10000000000uqor \
  --from qor1abc...def
```

5.7 Privacy and Data Protection

QCAI processes transaction data and contract code, both of which may contain sensitive information. The privacy architecture enforces the following guarantees:

1. **No Data Retention:** Off-chain QCAI inference requests are processed in ephemeral compute environments. Input data is purged immediately after the response is generated.

2. **On-Chain Anonymization:** On-chain models operate on feature vectors derived from transaction data, not raw addresses or amounts. The feature extraction function ϕ is designed to be non-invertible:

$$\phi : \mathcal{T} \to \mathbb{R}^d, \quad \text{s.t. } \nexists\, \phi^{-1} \text{ that recovers } \mathcal{T} \text{ from } \phi(\mathcal{T}) \tag{5.42}$$

3. **Differential Privacy:** Aggregated analytics produced by QCAI satisfy (ϵ, δ)-differential privacy. For any two adjacent datasets D, D' differing in a single transaction, and any output set S:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} \cdot \Pr[\mathcal{M}(D') \in S] + \delta \tag{5.43}$$

   where $\epsilon = 1.0$ and $\delta = 10^{-5}$ are the privacy parameters governing the noise calibration of the Gaussian mechanism applied to query results.

Chapter 6

Smart Contract Platform and Triple-VM Architecture

6.1 Unified Execution Environment

QoreChain implements a triple virtual machine architecture that hosts three distinct execution environments within a single blockchain: the Ethereum Virtual Machine (EVM), CosmWasm, and the Solana Virtual Machine (SVM). Rather than operating as isolated sidechains or rollups, these three VMs share a unified state layer and a common transaction ordering, enabling atomic cross-VM interactions without bridges or relay mechanisms.

6.1.1 Design Philosophy

The triple-VM architecture addresses a fundamental fragmentation problem in the blockchain ecosystem. Developers are forced to choose a single execution environment, inheriting its limitations alongside its strengths. QoreChain eliminates this tradeoff by providing:

1. **EVM**: The largest smart contract ecosystem with mature tooling, Solidity/Vyper support, and access to audited DeFi primitives.
2. **CosmWasm**: Rust-native contracts with the actor model, IBC interoperability, and memory-safe execution.
3. **SVM**: BPF-based parallel execution for high-throughput applications requiring Solana-compatible program deployment.

Motivation for triple-VM: The historical evolution of blockchains created a fragmented ecosystem where each chain optimized for different use cases. Ethereum became the DeFi standard through network effects and mature tooling; Cosmos developed IBC for cross-chain communication; Solana achieved high throughput through parallel execution. Rather than replicating only one model or forcing developers to make binary choices, QoreChain's architecture treats all three as co-equal execution tiers. This allows deploying a DEX on EVM (leveraging mature Uniswap-compatible tooling), a cross-chain bridge on CosmWasm (using native IBC support), and a high-throughput trading engine on SVM (using parallel account execution) within the same transaction domain, with atomic settlement across all three VMs.

Execution model unification: Despite their distinct semantics, all three VMs share a unified transaction ordering, block boundaries, and finality guarantees. A transaction targeting multiple VMs executes atomically; either all VM-specific components succeed or all fail, with no partial state mutations. This atomicity is enforced through a cross-VM dispatcher that sequences transactions across VMs and validates state consistency at block boundaries using the shared state layer. The unified ordering prevents race conditions (e.g., a DEX trade and oracle price update on different VMs that could cause MEV if ordered independently) and ensures that developers can reason about transaction semantics globally rather than per-VM.

Developer productivity and ecosystem lock-in: By supporting all three ecosystems simultaneously, QoreChain captures network effects from existing Ethereum, Cosmos, and Solana developer communities. An Ethereum developer can deploy Solidity contracts without learning Rust or Rust assembly; a Cosmos builder can access Solana's throughput without reimplementing core logic. This reduces barriers to cross-ecosystem innovation and creates an ecosystem lock-in advantage: once developers have deployed to multiple VMs on QoreChain and benefited from atomic interoperability, migrating to a single-VM chain becomes economically unattractive.

6.1.2 Formal State Model

Let $\mathcal{S}$ denote the global blockchain state, which is partitioned into VM-specific substates:

$$\mathcal{S} = \mathcal{S}_{\text{EVM}} \cup \mathcal{S}_{\text{CW}} \cup \mathcal{S}_{\text{SVM}} \cup \mathcal{S}_{\text{shared}} \tag{6.1}$$

where $\mathcal{S}_{\text{shared}}$ contains cross-VM state (account balances, token mappings, cross-VM call records). The global state transition function δ processes a block of transactions $\mathbf{T} = (t_1, t_2, \ldots, t_n)$:

$$\mathcal{S}' = \delta(\mathcal{S}, \mathbf{T}) = \delta_n \circ \delta_{n-1} \circ \cdots \circ \delta_1(\mathcal{S}) \tag{6.2}$$

where each δ_i is the state transition induced by transaction t_i. For a transaction targeting VM $v \in \{\text{EVM}, \text{CW}, \text{SVM}\}$, the transition is:

$$\delta_i(\mathcal{S}) = \begin{cases} \delta_{\text{EVM}}(\mathcal{S}_{\text{EVM}}, \mathcal{S}_{\text{shared}}, t_i) & \text{if target}(t_i) = \text{EVM} \\ \delta_{\text{CW}}(\mathcal{S}_{\text{CW}}, \mathcal{S}_{\text{shared}}, t_i) & \text{if target}(t_i) = \text{CW} \\ \delta_{\text{SVM}}(\mathcal{S}_{\text{SVM}}, \mathcal{S}_{\text{shared}}, t_i) & \text{if target}(t_i) = \text{SVM} \\ \delta_{\text{cross}}(\mathcal{S}, t_i) & \text{if } t_i \text{ is cross-VM} \end{cases} \tag{6.3}$$

Theorem 1 (State Consistency)**.** *For any sequence of transactions* $\mathbf{T}$*, the global state transition preserves the balance invariant:*

$$\sum_{a \in \mathcal{A}} \mathit{bal}(a, \mathcal{S}') = \sum_{a \in \mathcal{A}} \mathit{bal}(a, \mathcal{S}) - \mathit{fees}(\mathbf{T}) - \mathit{burned}(\mathbf{T}) \tag{6.4}$$

where $\mathcal{A}$ *is the set of all accounts across all three VMs,* $\mathit{fees}(\mathbf{T})$ *is the total fees collected, and* $\mathit{burned}(\mathbf{T})$ *is the total QOR burned per the fee distribution schedule.*

Proof sketch. Each individual VM transition δ_v preserves the local balance invariant within its substate. The shared state $\mathcal{S}_{\text{shared}}$ acts as a conservation layer: any transfer from $\mathcal{S}_v$ to $\mathcal{S}_w$ ($v \neq w$) is mediated through $\mathcal{S}_{\text{shared}}$ via a debit-credit mechanism that

atomically decrements the source and increments the destination. The fee distribution (37% validators, 30% burn, 20% treasury, 10% stakers, 3% light nodes) is applied as a final step in δ after all transactions are processed, ensuring the invariant holds at block boundaries. □ □

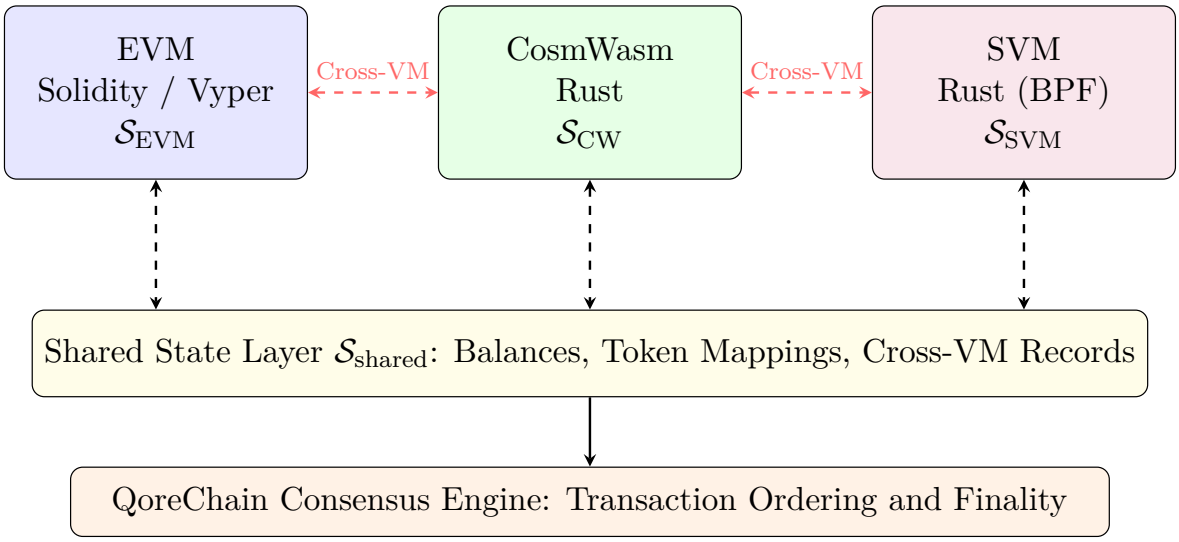

Figure 6.1: Triple-VM architecture with shared state layer

Querying VM Status and Metrics (CLI)

```
# List all active VMs and their status
qorchaind query vm status

# Get per-VM transaction throughput for the last 100 blocks
qorchaind query vm throughput --blocks 100

# Query the shared state layer for cross-VM token mappings
qorchaind query vm token-map --denom uqor
```

6.2 EVM Compatibility Layer

The QoreChain EVM provides full Ethereum equivalence: all standard opcodes, all nine Ethereum precompiled contracts, and JSON-RPC compatibility with existing Ethereum tooling (MetaMask, Hardhat, Foundry, Remix). Contracts deployed on Ethereum can be redeployed on QoreChain without modification.

6.2.1 PQC-Extended Precompiles

Beyond standard Ethereum precompiles, QoreChain introduces four post-quantum cryptographic precompiles accessible from Solidity at designated addresses:

Address	Precompile	Operation	Gas Cost
`0x0A01`	PQC_VERIFY	ML-DSA-87 signature verification	$G_{\text{verify}} = 12{,}000$
`0x0A02`	PQC_ENCAPSULATE	ML-KEM-1024 key encapsulation	$G_{\text{encap}} = 8{,}000$
`0x0A03`	PQC_DECAPSULATE	ML-KEM-1024 key decapsulation	$G_{\text{decap}} = 9{,}500$
`0x0A04`	SHAKE256	SHAKE-256 variable-length hash	$G_{\text{shake}} = 60 + 12 \cdot \lceil n/136 \rceil$

Table 6.1: QoreChain PQC precompiled contracts

The gas costs are calibrated against the standard `ecrecover` precompile (3,000 gas) to reflect the relative computational overhead of lattice-based operations. The SHAKE-256 gas formula accounts for the sponge construction's per-block processing cost, where n is the input length in bytes and 136 is the SHAKE-256 rate in bytes.

6.2.2 EVM State Transition Formalization

The EVM state $\mathcal{S}_{\text{EVM}}$ is defined as a mapping from 20-byte addresses to account tuples:

$$\mathcal{S}_{\text{EVM}} : \mathbb{B}_{20} \to (\text{nonce} \in \mathbb{N},\ \text{balance} \in \mathbb{N},\ \text{storageRoot} \in \mathbb{B}_{32},\ \text{codeHash} \in \mathbb{B}_{32}) \quad (6.5)$$

A transaction t triggers a state transition via the EVM execution function Ξ:

$$(\mathcal{S}'_{\text{EVM}}, g_{\text{remaining}}, \mathbf{o}, \mathbf{L}) = \Xi(\mathcal{S}_{\text{EVM}}, t) \quad (6.6)$$

where $g_{\text{remaining}}$ is the unused gas, $\mathbf{o}$ is the output data, and $\mathbf{L}$ is the set of emitted logs. The gas consumed is:

$$g_{\text{used}} = g_{\text{limit}} - g_{\text{remaining}} + g_{\text{refund\_}}cap \quad (6.7)$$

where the refund is capped at $g_{\text{refund\_cap}} = \lfloor g_{\text{used}}/5 \rfloor$ (following EIP-3529).

6.2.3 AI-Optimized Gas Estimation

The QCAI gas estimator (introduced in Chapter 5) provides improved gas predictions for EVM transactions. The standard estimation uses trace-based simulation, which is exact but slow. The AI estimator provides a fast approximation:

$$\hat{g}_{\text{AI}}(t) = f_\theta(\mathbf{x}_t) + \epsilon_{\text{safety}} \quad (6.8)$$

where $\mathbf{x}_t$ is a feature vector encoding the transaction's target contract, function selector, call data pattern, and historical gas usage for similar transactions. The safety margin ϵ_{safety} is computed from the model's prediction uncertainty:

$$\epsilon_{\text{safety}} = z_\alpha \cdot \hat{\sigma}_\theta(\mathbf{x}_t) \quad (6.9)$$

where z_α is the α-quantile of the standard normal (e.g., $z_{0.99} = 2.326$ for 99% confidence) and $\hat{\sigma}_\theta$ is the model's predicted standard deviation.

Using PQC Precompiles in Solidity

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

/// @title QoreChain PQC Utilities
/// @notice Library for interacting with QoreChain's PQC precompiles
library QorePQC {
    address constant PQC_VERIFY = address(0x0A01);
    address constant PQC_ENCAPSULATE = address(0x0A02);
    address constant PQC_DECAPSULATE = address(0x0A03);
    address constant SHAKE256 = address(0x0A04);

    /// @notice Verify an ML-DSA-87 signature
    /// @param message The signed message
    /// @param signature The ML-DSA-87 signature (4,627 bytes)
    /// @param publicKey The ML-DSA-87 public key (2,592 bytes)
    /// @return valid True if the signature is valid
    function verifyMLDSA(
        bytes memory message,
        bytes memory signature,
        bytes memory publicKey
    ) internal view returns (bool valid){
        bytes memory input = abi.encodePacked(
            uint32(message.length), message,
            uint32(signature.length), signature,
            publicKey
        );
        (bool success, bytes memory result) =
            ↪ PQC_VERIFY.staticcall(input);
        return success && abi.decode(result, (bool));
    }

    /// @notice Compute SHAKE-256 hash with variable output length
    /// @param data Input data
    /// @param outputLen Desired output length in bytes
    /// @return hash The SHAKE-256 digest
    function shake256(
        bytes memory data,
        uint256 outputLen
    ) internal view returns (bytes memory hash){
        bytes memory input = abi.encodePacked(
            uint32(outputLen), data
        );
        (bool success, bytes memory result) =
            ↪ SHAKE256.staticcall(input);
        require(success, "SHAKE256 precompile failed");
        return result;
```

```
    }
}

/// @title PQC-Protected Vault
/// @notice Example vault using ML-DSA-87 for withdrawal
    ↪ authorization
contract PQCVault {
    using QorePQC for *;

    mapping(bytes32 => uint256) public balances;
    mapping(bytes32 => bytes) public registeredKeys; // keyHash =>
        ↪ pubkey

    event Deposit(bytes32 indexed keyHash, uint256 amount);
    event Withdrawal(bytes32 indexed keyHash, uint256 amount);

    /// @notice Register a PQC public key and deposit funds
    function depositWithKey(bytes calldata pqcPubKey) external
        ↪ payable {
        bytes32 keyHash = keccak256(pqcPubKey);
        if (registeredKeys[keyHash].length == 0){
            registeredKeys[keyHash] = pqcPubKey;
        }
        balances[keyHash] += msg.value;
        emit Deposit(keyHash, msg.value);
    }

    /// @notice Withdraw funds using ML-DSA-87 signature
    function withdraw(
        uint256 amount,
        bytes calldata pqcPubKey,
        bytes calldata pqcSignature
    ) external {
        bytes32 keyHash = keccak256(pqcPubKey);
        require(balances[keyHash] >= amount, "Insufficient balance");

        // Construct withdrawal message
        bytes memory message = abi.encodePacked(
            block.chainid, address(this), keyHash, amount,
                ↪ block.number
        );

        // Verify ML-DSA-87 signature via precompile
        require(
            QorePQC.verifyMLDSA(message, pqcSignature, pqcPubKey),
            "Invalid PQC signature"
        );

        balances[keyHash] -= amount;
        payable(msg.sender).transfer(amount);
```

```
        emit Withdrawal(keyHash, amount);
    }
}
```

Deploying an EVM Contract (SDK)

```
import { QoreChainClient, EvmUtils } from "@qorechain/sdk";
import { readFileSync } from "fs";

const client = new QoreChainClient({
  rpcEndpoint: "https://rpc.qorechain.io",
  signer: wallet,
});

// Deploy a compiled Solidity contract
const artifact =
    ↪ JSON.parse(readFileSync("./artifacts/PQCVault.json", "utf8"));

const deployTx = await client.evm.deploy({
  bytecode: artifact.bytecode,
  abi: artifact.abi,
  constructorArgs: [],
  gasLimit: "auto", // uses QCAI gas estimation
});

console.log(`Contract deployed at: ${deployTx.contractAddress}`);
console.log(`Tx hash: ${deployTx.txHash}`);
console.log(`Gas used: ${deployTx.gasUsed} (estimated:
    ↪ ${deployTx.gasEstimate})`);

// Interact with the deployed contract
const vault = client.evm.contract(deployTx.contractAddress,
    ↪ artifact.abi);
const depositTx = await vault.depositWithKey(pqcPublicKey, {
  value: EvmUtils.parseEther("10"),
});
console.log(`Deposit confirmed: ${depositTx.txHash}`);
```

6.3 CosmWasm Integration

CosmWasm provides a WebAssembly-based smart contract runtime following the actor model. Contracts are written in Rust, compiled to Wasm, and communicate exclusively through typed messages. This architecture eliminates reentrancy vulnerabilities by design and provides memory safety guarantees inherited from Rust's ownership system.

6.3.1 Actor Model Formalization

Each CosmWasm contract is modeled as a deterministic state machine (actor) $\mathcal{A}_i$ defined by the tuple:

$$\mathcal{A}_i = (S_i, \mathcal{M}_i^{\text{in}}, \mathcal{M}_i^{\text{out}}, \delta_i, s_0) \tag{6.10}$$

where S_i is the set of possible internal states, $\mathcal{M}_i^{\text{in}}$ is the set of accepted input messages, $\mathcal{M}_i^{\text{out}}$ is the set of output messages (including submessages to other contracts), $\delta_i : S_i \times \mathcal{M}_i^{\text{in}} \to S_i \times \mathcal{M}_i^{\text{out}*} \times \mathcal{R}$ is the transition function, and s_0 is the initial state after instantiation. $\mathcal{R}$ denotes the set of response attributes (events, data).

The execution of a message $m \in \mathcal{M}_i^{\text{in}}$ against actor $\mathcal{A}_i$ in state s produces:

$$(s', \mathbf{m}_{\text{sub}}, r) = \delta_i(s, m) \tag{6.11}$$

where $\mathbf{m}_{\text{sub}} = (m_1, m_2, \ldots, m_k) \in \mathcal{M}_i^{\text{out}*}$ is a sequence of submessages dispatched to other actors. Submessages are processed depth-first with configurable reply-on semantics:

$$\text{SubMsg}(m_j) = (m_j,\ \text{reply\_on} \in \{\text{success}, \text{error}, \text{always}, \text{never}\},\ \text{gas\_limit}) \tag{6.12}$$

Theorem 2 (Reentrancy Safety)**.** *The actor model transition function δ_i completes all state mutations to S_i before any submessage $m_j \in \mathbf{m}_{sub}$ is dispatched. Therefore, for any actor $\mathcal{A}_i$, no external call can observe $\mathcal{A}_i$ in a partially updated state, eliminating reentrancy as a vulnerability class.*

6.3.2 IBC-Native Interoperability

CosmWasm contracts on QoreChain have native access to the Inter-Blockchain Communication (IBC) protocol. A contract can send and receive packets across any IBC-connected chain without relying on external bridge infrastructure. The IBC channel model is formalized as:

$$\text{Channel}(A, B) = (\text{portID}_A, \text{channelID}_A, \text{portID}_B, \text{channelID}_B, \text{ordering}, \text{version}) \tag{6.13}$$

where ordering $\in \{\text{ORDERED}, \text{UNORDERED}\}$ determines whether packets must be processed sequentially or can be delivered out of order.

CosmWasm Contract: PQC-Enabled Escrow (Rust)

```rust
use cosmwasm_std::{
    entry_point, to_json_binary, Binary, Deps, DepsMut,
    Env, MessageInfo, Response, StdResult, Uint128, Addr,
    BankMsg, Coin, SubMsg, WasmMsg,
};
use schemars::JsonSchema;
use serde::{Deserialize, Serialize};

#[derive(Serialize, Deserialize, Clone, JsonSchema)]
pub struct InstantiateMsg {
    pub arbiter: String,
    pub seller: String,
    pub timeout_blocks: u64,
    pub platform_fee_bps: u64, // basis points (e.g., 100 = 1%)
}

#[derive(Serialize, Deserialize, Clone, JsonSchema)]
#[serde(rename_all = "snake_case")]
pub enum ExecuteMsg {
    /// Buyer deposits funds into escrow
    Deposit {},
    /// Seller claims funds after delivery confirmation
    Release { pqc_signature: Binary },
    /// Arbiter resolves a dispute
    Resolve { release_to_seller: bool },
    /// Auto-refund after timeout
    Refund {},
    /// Cross-VM call: verify PQC signature via EVM precompile
    VerifyPqcViaEvm { message: Binary, signature: Binary, pubkey:
        ↪ Binary },
}

#[derive(Serialize, Deserialize, Clone, JsonSchema)]
pub struct State {
    pub buyer: Addr,
    pub seller: Addr,
    pub arbiter: Addr,
    pub amount: Uint128,
    pub timeout_block: u64,
    pub platform_fee_bps: u64,
    pub released: bool,
}

#[entry_point]
pub fn instantiate(
    deps: DepsMut,
```

```
    env: Env,
    info: MessageInfo,
    msg: InstantiateMsg,
) -> StdResult<Response> {
    let state = State {
        buyer: info.sender.clone(),
        seller: deps.api.addr_validate(&msg.seller)?,
        arbiter: deps.api.addr_validate(&msg.arbiter)?,
        amount: Uint128::zero(),
        timeout_block: env.block.height + msg.timeout_blocks,
        platform_fee_bps: msg.platform_fee_bps,
        released: false,
    };
    deps.storage.set(b"state", &to_json_binary(&state)?);
    Ok(Response::new().add_attribute("action", "instantiate_escrow"))
}

#[entry_point]
pub fn execute(
    deps: DepsMut,
    env: Env,
    info: MessageInfo,
    msg: ExecuteMsg,
) -> StdResult<Response> {
    match msg {
        ExecuteMsg::Deposit {} => execute_deposit(deps, info),
        ExecuteMsg::Release { pqc_signature } =>
            execute_release(deps, env, info, pqc_signature),
        ExecuteMsg::Resolve { release_to_seller } =>
            execute_resolve(deps, info, release_to_seller),
        ExecuteMsg::Refund {} => execute_refund(deps, env),
        ExecuteMsg::VerifyPqcViaEvm { message, signature, pubkey } =>
            execute_cross_vm_verify(deps, message, signature, pubkey),
    }
}

fn execute_cross_vm_verify(
    _deps: DepsMut,
    message: Binary,
    signature: Binary,
    pubkey: Binary,
) -> StdResult<Response> {
    // Cross-VM call to EVM PQC precompile via the VM bridge
    let evm_call = SubMsg::new(WasmMsg::Execute {
        contract_addr: "qor1_vm_bridge_contract".to_string(),
        msg: to_json_binary(&serde_json::json!({
            "call_evm_precompile": {
                "precompile": "0x0A01",
                "input": {
                    "message": message,
```

```
                    "signature": signature,
                    "public_key": pubkey
                }
            }
        }))?,
        funds: vec![],
    });
    Ok(Response::new()
        .add_submessage(evm_call)
        .add_attribute("action", "cross_vm_pqc_verify"))
}
```

Deploying and Interacting with CosmWasm Contracts (CLI)

```
# Compile the contract (from the contract directory)
cargo wasm
# Optimize the Wasm binary
docker run --rm -v "$(pwd)":/code \
  cosmwasm/rust-optimizer:0.15.0

# Store the contract on-chain
qorchaind tx wasm store ./artifacts/pqc_escrow.wasm \
  --from buyer_wallet --gas auto --gas-adjustment 1.3

# Instantiate the contract
qorchaind tx wasm instantiate 42 \
  '{"arbiter":"qor1arb...","seller":"qor1sell...",
    "timeout_blocks":50000,"platform_fee_bps":100}' \
  --label "pqc-escrow-v1" \
  --admin qor1buyer... \
  --from buyer_wallet --gas auto

# Execute a deposit
qorchaind tx wasm execute qor1contract... \
  '{"deposit":{' \}}
  --amount 1000000000uqor \
  --from buyer_wallet

# Query the escrow state
qorchaind query wasm contract-state smart qor1contract... \
  '{"get_state":{'}}
```

6.4 SVM (Solana Virtual Machine) Integration

The QoreChain SVM provides a BPF-based execution environment compatible with Solana's program model. Programs compiled for Solana can be deployed on

QoreChain's SVM, accessing the same account model and parallel execution capabilities while benefiting from QoreChain's quantum-safe security and cross-VM interoperability.

6.4.1 Account Model

The SVM uses an account-based model distinct from the EVM's contract storage model. Each account is a tuple:

$$\text{Account}_{\text{SVM}} = (\text{owner} \in \mathbb{B}_{32},\ \text{lamports} \in \mathbb{N},\ \text{data} \in \mathbb{B}^*,\ \text{executable} \in \{0,1\}) \quad (6.14)$$

Programs (executable accounts) process instructions that reference a set of input accounts. The account ownership model enforces that only the owning program can modify an account's data:

$$\forall a \in \mathcal{A}_{\text{SVM}} :\ \text{modify}(a.\text{data}) \implies \text{caller} = a.\text{owner} \quad (6.15)$$

Unified account model across VMs: While the SVM maintains its native account semantics for compatibility with Solana programs, QoreChain provides a unified addressing and balance model that spans all three VMs. Each account is represented using Bech32 addressing (the Cosmos standard) and maintains a single balance sheet across all VMs. When an account holds QOR tokens, the balance is unified regardless of whether the tokens are accessed through EVM (as a fungible token contract), CosmWasm (as native module balance), or SVM (as lamports in an SVM-compatible account). This eliminates the fragmentation where Ethereum users have ETH, Cosmos users have native tokens, and Solana users have SOL; on QoreChain, the native currency (QOR) is truly universal.

Cross-VM account state and consensus: Accounts can be referenced from any VM. An EVM smart contract can read an SVM account's balance; a CosmWasm contract can validate SVM program authorization; an SVM program can call an EVM contract. These cross-VM reads and writes pass through the shared state layer, which guarantees that all VMs observe identical account state at block boundaries. Consensus validators verify that all three VMs applied state transitions consistently: if an EVM transaction transfers tokens to an account, that same account's balance is updated in the unified ledger visible to CosmWasm and SVM queries.

Account-scoped data privacy: The SVM's data model allows programs to define account-scoped storage that is not visible to other programs unless explicitly queried. This differs from Ethereum's global storage model (any contract can read any other contract's state) and provides stronger encapsulation. Combined with the ownership constraint, account-scoped data creates a clear security boundary: a program can only modify accounts it owns, and can only read data that accounts explicitly expose through query interfaces. This model is enforced through the sandboxed execution environment and is validated during static analysis before contract deployment.

6.4.2 Parallel Execution Model

The SVM achieves parallelism by analyzing account dependencies before execution. Given a set of transactions $\{t_1, \ldots, t_n\}$, the dependency graph $G = (V, E)$ is constructed where $V = \{t_1, \ldots, t_n\}$ and:

$$(t_i, t_j) \in E \iff \text{WriteSet}(t_i) \cap \text{AccessSet}(t_j) \neq \emptyset \ \vee \ \text{WriteSet}(t_j) \cap \text{AccessSet}(t_i) \neq \emptyset \tag{6.16}$$

where $\text{WriteSet}(t)$ is the set of accounts written by transaction t and $\text{AccessSet}(t) = \text{ReadSet}(t) \cup \text{WriteSet}(t)$. The maximum parallelism achievable is:

$$P_{\max} = \frac{|V|}{\text{length of longest path in } G} \tag{6.17}$$

The chromatic number $\chi(G)$ of the conflict graph determines the minimum number of sequential execution rounds:

$$\text{Rounds}_{\min} = \chi(G), \quad \text{Speedup} = \frac{|V|}{\chi(G)} \tag{6.18}$$

For typical DeFi workloads where most transactions touch distinct accounts, empirical analysis shows $\chi(G) \approx O(\sqrt{n})$, yielding near-linear speedup.

6.4.3 Account Reconciliation with Shared State

The SVM account model is reconciled with the shared state layer through a mapping function μ that translates between SVM account addresses and QoreChain's unified address space:

$$\mu : \mathbb{B}_{32}^{\text{SVM}} \leftrightarrow \text{Bech32}(\texttt{qor}, \mathbb{B}_{20}) \tag{6.19}$$

Token balances are synchronized at block boundaries:

$$\text{bal}_{\text{shared}}(a) = \text{bal}_{\text{native}}(\mu^{-1}(a)) + \sum_{v \in \{\text{EVM},\text{CW}\}} \text{wrapped}(a, v) \tag{6.20}$$

Interacting with SVM Programs (SDK)

```
import { QoreChainClient } from "@qorechain/sdk";
import { PublicKey, TransactionInstruction } from
    ↪ "@qorechain/svm-sdk";

const client = new QoreChainClient({
  rpcEndpoint: "https://rpc.qorechain.io",
  signer: wallet,
});

// Deploy an SVM program
const programBinary = readFileSync("./target/deploy/my_program.so");
const deployResult = await client.svm.deployProgram({
  binary: programBinary,
  programId: newProgramKeypair,
});
console.log(`SVM program deployed: ${deployResult.programId}`);

// Execute an SVM instruction
```

```
const instruction = new TransactionInstruction({
  programId: new PublicKey(deployResult.programId),
  keys: [
    { pubkey: userAccount, isSigner: true, isWritable: true },
    { pubkey: dataAccount, isSigner: false, isWritable: true },
  ],
  data: Buffer.from([/* instruction data */]),
});

const txResult = await client.svm.sendInstruction(instruction);
console.log(`SVM tx confirmed: ${txResult.txHash}`);
console.log(`Compute units used: ${txResult.computeUnits}`);
```

6.5 Cross-VM Interoperability Protocol

The cross-VM interoperability protocol enables atomic interactions between contracts deployed on different virtual machines. An EVM contract can invoke a CosmWasm contract, which can in turn call an SVM program, all within a single atomic transaction.

6.5.1 Cross-VM Call Semantics

A cross-VM call is formalized as a message m_{cross} with the following structure:

$$m_{\text{cross}} = (\text{source\_vm}, \text{target\_vm}, \text{caller}, \text{callee}, \text{payload}, \text{callback}, \text{gas\_limit}) \quad (6.21)$$

The execution follows a synchronous call-return pattern with atomic rollback:

1. The source VM serializes the call payload and delegates to the cross-VM dispatcher.
2. The dispatcher routes the call to the target VM, translating address formats and data encoding.
3. The target VM executes the call and returns the result (or an error).
4. On success, both VM state changes are committed atomically. On failure, all state changes (including the source VM's) are reverted.

6.5.2 Atomicity via Two-Phase Commit

Cross-VM atomicity is enforced through a two-phase commit protocol adapted for the blockchain context. For a cross-VM transaction t_{cross} involving VMs v_1 and v_2:

Phase 1 (Prepare):

$$v_1 : \mathcal{S}'_{v_1} = \delta_{v_1}^{\text{tentative}}(\mathcal{S}_{v_1}, t) \quad \text{(not yet committed)} \quad (6.22)$$

$$v_2 : \mathcal{S}'_{v_2} = \delta_{v_2}^{\text{tentative}}(\mathcal{S}_{v_2}, m_{\text{cross}}) \quad \text{(not yet committed)} \quad (6.23)$$

Phase 2 (Commit/Abort): If both succeed: $\mathcal{S} \leftarrow \mathcal{S}'_{v_1} \cup \mathcal{S}'_{v_2}$ (6.24)

If either fails: $\mathcal{S} \leftarrow \mathcal{S}$ (no change) (6.25)

Theorem 3 (Cross-VM Atomicity)**.** *For any cross-VM transaction t_{cross}, the resulting global state $\mathcal{S}'$ satisfies:*

$$\mathcal{S}' \in \{\mathcal{S}_{both\_committed},\ \mathcal{S}_{original}\} \tag{6.26}$$

That is, no intermediate state where one VM's changes are committed but the other's are not is observable at any block boundary.

Proof sketch. The two-phase commit is executed within a single block's state transition function δ. Since the QoreChain Consensus Engine provides single-block finality, there is no window during which a partial state could be observed by external queries. The tentative state changes are held in a write-ahead journal that is either fully applied or fully discarded before the block is finalized. □ □

6.5.3 Data Encoding Translation

Cross-VM calls require translation between different data encodings:

Source VM	Target VM	Encoding Translation	Address Translation
EVM	CosmWasm	ABI → JSON	hex → bech32
EVM	SVM	ABI → Borsh	hex → base58
CosmWasm	EVM	JSON → ABI	bech32 → hex
CosmWasm	SVM	JSON → Borsh	bech32 → base58
SVM	EVM	Borsh → ABI	base58 → hex
SVM	CosmWasm	Borsh → JSON	base58 → bech32

Table 6.2: Cross-VM data encoding and address translation matrix

Cross-VM Call: Solidity Calling CosmWasm

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;

/// @title Cross-VM interface for calling CosmWasm contracts
interface IVMBridge {
    /// @notice Call a CosmWasm contract from EVM
    /// @param contractAddr The bech32 address of the CosmWasm
        ↪ contract
    /// @param jsonMsg The JSON-encoded execute message
    /// @param coins Coins to send with the message (JSON array)
    /// @return response The JSON-encoded response from CosmWasm
    function callCosmWasm(
        string calldata contractAddr,
        bytes calldata jsonMsg,
```

```
        string calldata coins
    ) external returns (bytes memory response);

    /// @notice Call an SVM program from EVM
    /// @param programId The base58 program ID
    /// @param instructionData The Borsh-encoded instruction
    /// @return response The program's return data
    function callSVM(
        string calldata programId,
        bytes calldata instructionData
    ) external returns (bytes memory response);
}

/// @title DeFi Aggregator using all three VMs
contract CrossVMAggregator {
    IVMBridge public immutable vmBridge;

    constructor(address _vmBridge){
        vmBridge = IVMBridge(_vmBridge);
    }

    /// @notice Swap tokens using the best rate across all three VMs
    /// @dev Queries EVM DEX, CosmWasm DEX, and SVM DEX, executes on
        ↪ best
    function crossVMSwap(
        address tokenIn,
        uint256 amountIn,
        uint256 minAmountOut
    ) external returns (uint256 amountOut){
        // Step 1: Get quote from CosmWasm DEX
        bytes memory cwQuote = vmBridge.callCosmWasm(
            "qor1cosmwasm_dex_address",
            abi.encodePacked('{"get_quote":{"offer_asset":"',
                ↪ tokenIn,}}
                '","amount":"', amountIn, '"'),
            "[]"
        );

        // Step 2: Get quote from SVM DEX
        bytes memory svmQuote = vmBridge.callSVM(
            "SVMDexProgramId...",
            abi.encodePacked(uint8(1), tokenIn, amountIn) // GetQuote
                ↪ ix
        );

        // Step 3: Execute on the VM with the best rate
        // (simplified for illustration)
        require(amountOut >= minAmountOut, "Slippage exceeded");
        return amountOut;
    }
```

```
}
```

Cross-VM Call: CosmWasm Calling EVM (Rust)

```rust
use cosmwasm_std::{
    to_json_binary, Binary, DepsMut, Env, MessageInfo,
    Response, StdResult, SubMsg, WasmMsg,
};

/// Call an EVM contract from a CosmWasm contract
pub fn call_evm_contract(
    _deps: DepsMut,
    _env: Env,
    _info: MessageInfo,
    evm_contract: String,  // hex address, e.g. "0xAbC..."
    function_sig: String,  // e.g. "transfer(address,uint256)"
    abi_params: Binary,    // ABI-encoded parameters
) -> StdResult<Response> {
    // Route through the VM bridge contract
    let bridge_msg = WasmMsg::Execute {
        contract_addr: "qor1_vm_bridge_contract".to_string(),
        msg: to_json_binary(&serde_json::json!({
            "call_evm": {
                "contract": evm_contract,
                "function": function_sig,
                "params": abi_params.to_base64(),
                "value": "0"
            }
        }))?,
        funds: vec![],
    };

    Ok(Response::new()
        .add_submessage(SubMsg::reply_on_success(bridge_msg, 1))
        .add_attribute("action", "cross_vm_evm_call")
        .add_attribute("target", evm_contract))
}
```

6.6 Unified Gas and Resource Metering

The three VMs use different native resource units: the EVM uses gas, CosmWasm uses gas (with different cost tables), and the SVM uses compute units. QoreChain normalizes these into a common unit called **QoreGas** ($\mathcal{G}_Q$) for unified fee calculation and cross-VM gas limits.

6.6.1 Normalization Functions

The normalization maps each VM's native unit to QoreGas:

$$\mathcal{G}_Q^{\text{EVM}}(g) = \alpha_{\text{EVM}} \cdot g + \beta_{\text{EVM}} \tag{6.27}$$

$$\mathcal{G}_Q^{\text{CW}}(g) = \alpha_{\text{CW}} \cdot g + \beta_{\text{CW}} \tag{6.28}$$

$$\mathcal{G}_Q^{\text{SVM}}(c) = \alpha_{\text{SVM}} \cdot c + \beta_{\text{SVM}} \tag{6.29}$$

where α_v is the per-unit scaling factor and β_v is a fixed overhead for VM context switching. The scaling factors are calibrated so that equivalent operations across VMs cost approximately the same in QoreGas:

$$\alpha_{\text{EVM}} \cdot g_{\text{SSTORE}} \approx \alpha_{\text{CW}} \cdot g_{\text{db\_write}} \approx \alpha_{\text{SVM}} \cdot c_{\text{account\_write}} \tag{6.30}$$

The fee in QOR for a transaction consuming $\mathcal{G}_Q$ QoreGas units is:

$$\text{Fee}_{\text{QOR}} = \mathcal{G}_Q \cdot P_{\text{QoreGas}} \cdot (1 + \Delta_{\text{congestion}}) \tag{6.31}$$

where P_{QoreGas} is the base price per QoreGas unit (set by the fee market) and $\Delta_{\text{congestion}}$ is the QCAI congestion multiplier from the predictive fee model (Section 5.2).

For cross-VM transactions, the total gas is the sum of gas consumed in each VM plus the translation overhead:

$$\mathcal{G}_Q^{\text{cross}} = \mathcal{G}_Q^{v_1}(g_1) + \mathcal{G}_Q^{v_2}(g_2) + \mathcal{G}_Q^{\text{translate}}(|\text{payload}|) \tag{6.32}$$

where $\mathcal{G}_Q^{\text{translate}}$ accounts for data encoding conversion costs that scale linearly with payload size.

Querying Cross-VM Gas Estimates (SDK)

```
import { QoreChainClient } from "@qorechain/sdk";

const client = new QoreChainClient({ rpcEndpoint:
    ↪ "https://rpc.qorechain.io" });

// Estimate gas for a cross-VM transaction
const gasEstimate = await client.vm.estimateCrossVMGas({
  sourceVM: "evm",
  targetVM: "cosmwasm",
  sourceContract: "0xAbC...123",
  targetContract: "qor1cosmwasm_addr...",
  payload: encodedCalldata,
});

console.log(`EVM gas: ${gasEstimate.sourceGas}`);
console.log(`CosmWasm gas: ${gasEstimate.targetGas}`);
console.log(`Translation overhead: ${gasEstimate.translationGas}`);
console.log(`Total QoreGas: ${gasEstimate.totalQoreGas}`);
console.log(`Estimated fee: ${gasEstimate.feeQOR} uqor`);

// Compare gas costs across VMs for the same operation
```

```
const comparison = await client.vm.compareGas({
  operation: "token_transfer",
  amount: "1000000uqor",
});
for (const vm of comparison.vms){
  console.log(`${vm.name}: ${vm.nativeGas} native, ${vm.qoreGas}
      ↪ QoreGas`);
}
```

6.7 Security Model

6.7.1 Sandboxed Execution

Each VM operates within a sandboxed execution environment that enforces strict isolation:

1. **Memory Isolation**: Each VM has its own memory space. The EVM uses a 256-bit word-addressable memory, CosmWasm uses linear Wasm memory bounded by configurable limits, and the SVM uses account-scoped data regions.

2. **State Isolation**: Direct cross-VM state access is prohibited. All cross-VM communication passes through the shared state layer and the cross-VM dispatcher, which validates permissions and enforces gas limits.

3. **Deterministic Execution**: All three VMs produce deterministic outputs for identical inputs, which is verified during consensus. Floating-point operations are either banned (EVM, CosmWasm) or replaced with fixed-point equivalents (SVM).

Execution resource limits: Each transaction specifies gas limits that apply independently per VM. An EVM component of a cross-VM transaction has a separate gas budget from its CosmWasm and SVM components; exhausting gas in one VM does not affect the others. This prevents a denial-of-service attack where a contract could starve other VMs of computational resources. Validators enforce these per-VM gas budgets during transaction execution, and all three VMs record gas consumption in their transaction receipts for fee accounting.

Memory bounds and buffer overflow prevention: CosmWasm contracts execute within Wasm memory sandboxes with fixed upper bounds (typically 1 GB per contract instance). The EVM's memory is similarly bounded through the gas cost model: accessing high memory addresses incurs quadratic gas costs, making unbounded memory exhaustion economically infeasible. The SVM's account-scoped data model prevents arbitrary memory access; programs can only manipulate data associated with accounts they own. All three sandboxes are enforced by the runtime before bytecode execution, ensuring that malicious or buggy contracts cannot corrupt memory belonging to other contracts or the kernel.

I/O and side-effect isolation: No smart contract can directly perform I/O operations (network, file system, random number generation). All external state mutations

pass through the kernel's controlled interfaces. Random number generation uses the blockchain's commit-reveal scheme, ensuring determinism for consensus. Network calls must go through explicit cross-chain bridges that record all mutations in the transaction log. This isolation prevents contracts from taking non-deterministic actions that could cause consensus failure if different validators observed different I/O outcomes.

Enforcement and verification: The isolation properties are enforced at three levels. First, at compilation: the contract compiler (Solidity-to-EVM, Rust-to-Wasm, Rust-to-BPF) enforces memory and I/O constraints by refusing to compile unsafe operations. Second, at deployment: validators run static analysis to verify that contracts do not attempt to circumvent the sandbox. Third, at runtime: the VM's memory management unit and permission checking system prevent policy violations in real time. If a contract attempts to access invalid memory or invoke unauthorized system calls, execution traps immediately and the transaction is reverted.

6.7.2 Abstract Interpretation for Static Analysis

QoreChain employs abstract interpretation to statically analyze smart contracts before deployment. Given a concrete program semantics $[\![P]\!] : \mathcal{S} \to \mathcal{S}$, the abstract interpretation computes an over-approximation $[\![P]\!]^\sharp : \mathcal{S}^\sharp \to \mathcal{S}^\sharp$ in an abstract domain $\mathcal{S}^\sharp$.

The soundness property guarantees that the analysis never misses a real vulnerability:

$$\forall s \in \mathcal{S} : \ \gamma([\![P]\!]^\sharp(\alpha(s))) \supseteq \{[\![P]\!](s)\} \tag{6.33}$$

where $\alpha : \mathcal{S} \to \mathcal{S}^\sharp$ is the abstraction function and $\gamma : \mathcal{S}^\sharp \to \mathcal{P}(\mathcal{S})$ is the concretization function, forming a Galois connection:

$$\alpha(s) \sqsubseteq s^\sharp \iff s \in \gamma(s^\sharp) \tag{6.34}$$

QoreChain's analyzer uses the interval abstract domain for numerical properties (detecting integer overflows), the pointer abstract domain for aliasing analysis (detecting storage collisions), and a custom taint domain for tracking untrusted data flows (detecting injection vulnerabilities).

Running Static Analysis on a Contract (CLI)

```
# Analyze a Solidity contract for vulnerabilities
qorchaind tx vm analyze-contract ./MyContract.sol \
  --vm evm \
  --analysis-depth full \
  --output findings.json

# Analyze a CosmWasm contract (Wasm binary)
qorchaind tx vm analyze-contract ./my_contract.wasm \
  --vm cosmwasm \
  --check reentrancy,overflow,unauthorized_access

# View analysis results
qorchaind query vm analysis-report \
```

```
  --contract qor1contract_addr... \
  --format table
```

Chapter 7

Multi-Layer Architecture and Rollup Development Kit

7.1 Architecture Overview

QoreChain's multi-layer architecture enables workload-specific scaling by distributing computation across four distinct layer types: sidechains, paychains, rollups, and hybrid chain state (HCS) instances. Each layer type settles to the QoreChain main chain, inheriting its quantum-safe security guarantees while operating with independent execution parameters optimized for its target workload.

The multi-layer module maintains a registry of all active subsidiary layers:

$$\mathcal{L} = \{(l_i, \tau_i, S_i, \text{anchor}_i) \mid i = 1, \ldots, N\} \tag{7.1}$$

where l_i is the layer identifier, $\tau_i \in \{\text{sidechain}, \text{paychain}, \text{rollup}, \text{hcs}\}$ is the layer type, S_i is the current state commitment, and anchor_i is the most recent main chain block height at which the layer's state was anchored.

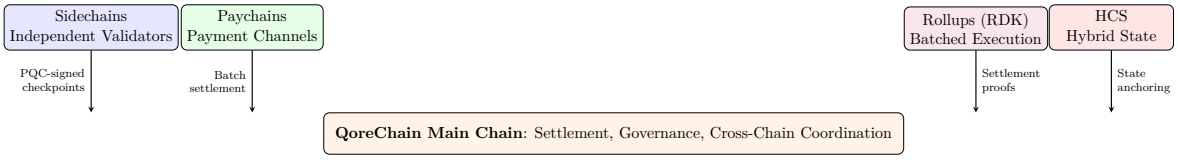

Figure 7.1: QoreChain multi-layer hierarchy with four layer types

7.1.1 Layer Type Comparison

Property	Sidechain	Paychain	Rollup	HCS
Consensus	Independent validator set	Probabilistic, batch settle	Sequencer + L1 proof	Hybrid on/off-chain
Finality	Sidechain finality + anchor	Instant (probabilistic)	Mode-dependent	Anchor-dependent
Security model	Own validators + main anchor	Main chain at settlement	Inherits main chain	PQC state anchoring
Throughput	High (dedicated resources)	Very high (channel-based)	Very high (batched)	Configurable
Ideal workload	Compute-heavy contracts	Micropayments, streaming	General-purpose scaling	Enterprise workflows

Table 7.1: Comparison of QoreChain layer types

7.1.2 Hierarchical State Anchoring

All four layer types anchor their state to the main chain using the Hierarchical Commitment Scheme (HCS protocol). The commitment for layer l_i at height h is:

$$\text{Commit}(l_i, h) = H_{\text{SHAKE-256}}(S_i^h \| h \| l_i \| \tau_i \| \text{nonce}) \tag{7.2}$$

The main chain maintains an aggregated commitment tree that enables efficient verification of any individual layer's state:

$$\text{AggTree}(h) = \text{MerkleRoot}\left(\{\text{Commit}(l_i, h)\}_{i=1}^{N}\right) \tag{7.3}$$

Verification of a single layer's state commitment requires $O(\log N)$ proof elements, where N is the number of active layers. This allows light clients and cross-layer contracts to verify subsidiary layer state without downloading the full aggregated tree.

Querying Multi-Layer Status (CLI)

```
# List all active layers and their types
qorchaind query multilayer list-layers

# Get detailed status for a specific layer
qorchaind query multilayer layer-info --layer-id "defi-rollup-01"

# Query the aggregated commitment tree at a given height
qorchaind query multilayer commitment-tree --height 100000

# Verify a layer's state commitment against the main chain
qorchaind query multilayer verify-commitment \
  --layer-id "defi-rollup-01" \
  --state-root "abc123..." \
  --height 100000
```

7.2 Rollup Development Kit (RDK)

The Rollup Development Kit (RDK) provides a framework for deploying application-specific rollups on QoreChain. The RDK abstracts the complexity of rollup infrastructure into configurable profiles, enabling developers to launch a rollup with production-grade settlement, data availability, and sequencing in a single transaction.

7.2.1 Rollup Configuration Space

An RDK rollup is fully specified by a configuration tuple:

$$\text{RollupConfig} = (\rho, \sigma, \pi, \delta, \gamma) \tag{7.4}$$

where:

- $\rho \in \{\text{defi}, \text{gaming}, \text{nft}, \text{enterprise}, \text{custom}\}$ is the rollup profile

- $\sigma \in \{\text{optimistic}, \text{zk}, \text{based}, \text{sovereign}\}$ is the settlement mode
- $\pi \in \{\text{fraud}, \text{snark}, \text{stark}, \text{none}\}$ is the proof system
- $\delta \in \{\text{dedicated}, \text{shared}, \text{based}\}$ is the sequencer mode
- $\gamma = (G_{\text{model}}, G_{\text{limit}}, G_{\text{price}})$ is the gas configuration

The configuration must satisfy a compatibility matrix $\mathcal{M}$ that enforces valid combinations. For example, ZK settlement requires either SNARK or STARK proof systems, and optimistic settlement requires fraud proofs:

$$\mathcal{M}(\sigma, \pi) = \begin{cases} \text{valid} & \text{if } (\sigma = \text{optimistic} \wedge \pi = \text{fraud}) \\ \text{valid} & \text{if } (\sigma = \text{zk} \wedge \pi \in \{\text{snark}, \text{stark}\}) \\ \text{valid} & \text{if } (\sigma = \text{based} \wedge \pi \in \{\text{fraud}, \text{snark}, \text{stark}\}) \\ \text{valid} & \text{if } (\sigma = \text{sovereign} \wedge \pi = \text{none}) \\ \text{invalid} & \text{otherwise} \end{cases} \tag{7.5}$$

7.2.2 Pre-Configured Rollup Profiles

The RDK provides five optimized profiles, each pre-tuned for a specific workload pattern:

Profile	Settlement	Proof	Sequencer	VM	Block Time	DA Backend	Gas Model
DeFi	ZK	SNARK	Dedicated	EVM	500 ms	Native	EIP-1559
Gaming	Based	Fraud	Dedicated	Custom	200 ms	Native	Flat fee
NFT	Optimistic	Fraud	Shared	CosmWasm	2 s	Celestia	Tiered
Enterprise	Based	STARK	Dedicated	EVM	1 s	Native	Subsidized
Custom	(user-defined)	(user-defined)	(user-defined)	(any)	(user-defined)	(any)	(user-defined)

Table 7.2: RDK pre-configured rollup profiles

The DeFi profile optimizes for value security with ZK proofs providing immediate settlement finality, which is critical for composable DeFi protocols where pending finality creates systemic risk. The Gaming profile prioritizes latency (200 ms block times) with based sequencing for censorship resistance, accepting the longer optimistic finality window since gaming transactions are typically lower value. The NFT profile leverages Celestia for data availability, reducing on-chain storage costs for media-heavy metadata. The Enterprise profile uses STARK proofs for transparency (no trusted setup) combined with subsidized gas to support enterprise adoption.

7.2.3 Rollup State Transition Function

For a rollup R with state S_R, the state transition function processes a batch of transactions $\mathbf{T}_b = (t_1, \ldots, t_m)$ in batch b:

$$S_R^{b+1} = \Delta_R(S_R^b, \mathbf{T}_b) \tag{7.6}$$

The batch submission to the main chain includes:

$$\text{BatchSubmission}(b) = \left(R_{\text{id}}, b, S_R^b, S_R^{b+1}, \text{txRoot}(\mathbf{T}_b), |\mathbf{T}_b|, \pi_b, \sigma_{\text{seq}}\right) \tag{7.7}$$

where π_b is the proof (type depends on settlement mode), σ_{seq} is the sequencer's PQC signature, and txRoot is the Merkle root of the transaction batch.

The compression ratio achieved by rollup batching is:

$$\text{CR}(b) = \frac{\sum_{i=1}^{m} |t_i|}{|\text{BatchSubmission}(b)|} = \frac{m \cdot \bar{t}}{|S_R^b| + |S_R^{b+1}| + 32 + |\pi_b| + |\sigma_{\text{seq}}|} \tag{7.8}$$

where $\bar{t}$ is the average transaction size. For typical DeFi batches ($m = 1000$, $\bar{t} = 256$ bytes), the compression ratio exceeds 100x for optimistic settlement and 50x for ZK settlement (due to the larger proof π_b).

Creating and Managing Rollups (CLI)

```
# Create a rollup using the DeFi profile
qorchaind tx rdk create-rollup \
  --profile defi \
  --name "QoreDEX Rollup" \
  --stake 10000000000uqor \
  --from qor1creator...

# Create a custom rollup with explicit configuration
qorchaind tx rdk create-rollup \
  --profile custom \
  --settlement-mode zk \
  --proof-system stark \
  --sequencer-mode shared \
  --da-backend both \
  --block-time 1s \
  --gas-model eip1559 \
  --name "Enterprise Settlement Layer" \
  --stake 10000000000uqor \
  --from qor1creator...

# List all active rollups
qorchaind query rdk list-rollups --status active

# Get rollup details and settlement history
qorchaind query rdk rollup --rollup-id "qoredex-rollup-01"

# Pause a rollup (creator only)
qorchaind tx rdk pause-rollup \
  --rollup-id "qoredex-rollup-01" \
  --from qor1creator...
```

7.3 Settlement Modes

7.3.1 Optimistic Settlement

Optimistic rollups assume batch validity by default and rely on a challenge mechanism to detect fraud. After a batch is submitted, a challenge window W_c (default: 7 days) opens during which any network participant can submit a fraud proof.

Fraud Proof Game Theory. The challenge mechanism is modeled as a two-player game between the sequencer (proposer) and potential challengers. Let V_b denote the

total value of transactions in batch b. The sequencer posts a bond B_s and the challenger posts a bond B_c.

The payoff matrix for an honest batch:

$$\text{Payoff}_{\text{honest}} = \begin{cases} \text{Sequencer :} & +\text{fees}(b) - \text{gas}_{\text{submit}} \\ \text{Challenger (no challenge) :} & 0 \\ \text{Challenger (false challenge) :} & -B_c \quad \text{(bond slashed)} \end{cases} \tag{7.9}$$

The payoff matrix for a fraudulent batch:

$$\text{Payoff}_{\text{fraud}} = \begin{cases} \text{Sequencer (unchallenged) :} & +V_b + \text{fees}(b) \\ \text{Sequencer (challenged) :} & -B_s \quad \text{(bond slashed)} \\ \text{Challenger (valid challenge) :} & +B_s/2 \quad \text{(reward from sequencer bond)} \end{cases} \tag{7.10}$$

Theorem 4 (Optimistic Security)**.** *If the sequencer bond satisfies $B_s > V_b$ and at least one rational challenger exists with cost of verification $c_v < B_s/2$, then the unique Nash equilibrium is (honest submission, no challenge). That is, the sequencer has no incentive to submit fraudulent batches.*

Proof sketch. Consider the sequencer's decision. If submitting a fraudulent batch, the expected payoff is:

$$\mathbb{E}[\text{fraud}] = (1 - p_c)(V_b + \text{fees}) + p_c(-B_s) = V_b + \text{fees} - p_c(V_b + \text{fees} + B_s) \tag{7.11}$$

where p_c is the probability of being challenged. A rational challenger challenges whenever $B_s/2 > c_v$, so $p_c = 1$ when this condition holds. Substituting:

$$\mathbb{E}[\text{fraud}] = V_b + \text{fees} - (V_b + \text{fees} + B_s) = -B_s < 0 \tag{7.12}$$

Since the honest payoff is $+\text{fees} - \text{gas} > 0$, honest submission strictly dominates. □ □

The batch lifecycle under optimistic settlement follows:

$$\text{submitted} \xrightarrow{W_c \text{ expires, no challenge finalized}} \qquad \text{submitted} \xrightarrow{\text{valid fraud proof}} \text{rejected} \tag{7.13}$$

Auto-finalization occurs at the end of each block's EndBlocker: batches whose challenge window has expired are promoted to finalized status.

7.3.2 ZK Settlement

ZK (zero-knowledge) rollups provide immediate settlement finality by submitting a cryptographic validity proof alongside each batch. The main chain verifier checks:

$$\text{Verify}(\text{vk}, (S_R^b, S_R^{b+1}, \text{txRoot}), \pi_b) \stackrel{?}{=} \text{accept} \tag{7.14}$$

where vk is the verification key registered at rollup creation and π_b is the SNARK or STARK proof. Upon successful verification, the batch is immediately finalized with no challenge window.

Property	SNARK	STARK	Implication
Trusted setup	Required	Not required	STARKs preferred for trust minimization
Proof size	$O(1)$ ($\sim$200 bytes)	$O(\log^2 n)$ ($\sim$50 KB)	SNARKs lower on-chain cost
Prover time	$O(n \log n)$	$O(n \cdot \text{polylog}(n))$	Comparable in practice
Verifier time	$O(1)$	$O(\log^2 n)$	SNARKs faster verification
Quantum safety	Vulnerable (pairing-based)	Safe (hash-based)	STARKs align with PQC mission

Table 7.3: SNARK vs. STARK proof system comparison

SNARK vs. STARK Tradeoffs. For QoreChain's quantum-safe mission, STARK proofs are the recommended default for ZK rollups, as they rely on hash-based commitments (compatible with SHAKE-256) rather than elliptic curve pairings that are vulnerable to quantum attack. However, SNARKs remain available for use cases where proof size and verification cost are critical constraints.

The gas cost of on-chain ZK proof verification is:

$$G_{\text{zk}} = G_{\text{base}} + G_{\text{per\_element}} \cdot |\pi_b| + G_{\text{pairing}} \cdot n_{\text{pairings}} \tag{7.15}$$

where $n_{\text{pairings}} = 0$ for STARKs (no pairing operations).

7.3.3 Based Settlement

Based rollups delegate transaction sequencing to the QoreChain main chain validators. Transactions are submitted directly to the L1 mempool and ordered by the main chain's consensus, providing censorship resistance equivalent to the main chain itself.

The sequencing function for a based rollup R is:

$$\text{Seq}_{\text{based}}(R, h) = \{tx \in \text{Block}(h) \mid \text{target}(tx) = R\} \tag{7.16}$$

That is, the rollup's transaction sequence at height h is simply the subset of main chain block h that targets rollup R. This eliminates the need for a dedicated sequencer but introduces latency equal to the main chain block time.

Based settlement can use either fraud proofs or validity proofs:

$$\text{BasedFinality}(b) = \begin{cases} \text{MainChainFinality} + W_c & \text{if proof} = \text{fraud} \\ \text{MainChainFinality} + \text{ProveTime} & \text{if proof} \in \{\text{snark}, \text{stark}\} \end{cases} \tag{7.17}$$

7.3.4 Sovereign Settlement

Sovereign rollups use QoreChain exclusively for data availability, maintaining their own consensus and settlement rules. The main chain stores DA blobs but does not verify state transitions:

$$\text{SovereignCommit}(b) = \text{DACommitment}\left(H_{\text{SHA}} - 256(\text{blob}_b), |\text{blob}_b|, h\right) \tag{7.18}$$

Sovereign rollups are responsible for their own security guarantees. They are suited for chains that want to leverage QoreChain's quantum-safe DA layer while maintaining full sovereignty over execution rules and governance.

Rollup Settlement Operations (CLI)

```
# Submit a settlement batch (optimistic)
qorchaind tx rdk submit-batch \
  --rollup-id "qoredex-rollup-01" \
  --pre-state-root "abc123..." \
  --post-state-root "def456..." \
  --tx-count 1000 \
  --proof-type fraud \
  --from qor1sequencer...

# Challenge a batch (within challenge window)
qorchaind tx rdk challenge-batch \
  --rollup-id "qoredex-rollup-01" \
  --batch-index 42 \
  --fraud-proof ./fraud_proof.bin \
  --bond 5000000000uqor \
  --from qor1challenger...

# Query batch status
qorchaind query rdk batch \
  --rollup-id "qoredex-rollup-01" \
  --batch-index 42

# Get latest finalized batch
qorchaind query rdk latest-batch \
  --rollup-id "qoredex-rollup-01" \
  --status finalized
```

Rollup Settlement SDK (JavaScript)

```
import { QoreChainClient } from "@qorechain/sdk";

const client = new QoreChainClient({
  rpcEndpoint: "https://rpc.qorechain.io",
  signer: sequencerWallet,
});

// Submit a ZK settlement batch with STARK proof
const batchResult = await client.rdk.submitBatch({
  rollupId: "enterprise-layer-01",
  preStateRoot: preRoot,
  postStateRoot: postRoot,
  txCount: 500,
  proofType: "stark",
  proof: starkProofBytes,
});
```

```
console.log(`Batch ${batchResult.batchIndex} submitted`);
console.log(`Status: ${batchResult.status}`); // "finalized" for ZK
console.log(`Settlement tx: ${batchResult.txHash}`);

// Monitor batch finalization (for optimistic batches)
const subscription = client.rdk.subscribeBatchStatus({
  rollupId: "qoredex-rollup-01",
  batchIndex: 42,
});
subscription.on("statusChange", (status) => {
  console.log(`Batch 42 status: ${status}`);
  // submitted -> challenged -> finalized/rejected
});
```

7.4 Data Availability

Every rollup batch requires its transaction data to be available so that any participant can reconstruct the rollup state and verify (or challenge) settlement claims. The RDK provides two data availability backends.

7.4.1 Native DA

The native DA backend stores data directly on the QoreChain main chain. Each DA blob is committed using SHA-256:

$$\text{NativeCommit}(\text{blob}) = (H_{\text{SHA}} - 256(\text{blob}), |\text{blob}|, h_{\text{stored}}) \tag{7.19}$$

Native DA provides the strongest availability guarantee (same liveness as the main chain) but incurs higher storage costs. Blobs are stored in the RDK module's KV store and pruned after the retention period T_{retain} (default: 432,000 blocks, approximately 30 days at 6-second block times):

$$\text{Prune}(\text{blob}) \iff h_{\text{current}} - h_{\text{stored}} > T_{\text{retain}} \tag{7.20}$$

The maximum blob size is 2 MB per submission, with a maximum of 10 batches per block, yielding a theoretical DA throughput of:

$$\text{DA}_{\text{max}} = \frac{10 \times 2 \text{ MB}}{6 \text{ s}} = 3.33 \text{ MB/s} \tag{7.21}$$

7.4.2 Celestia DA (via IBC)

For rollups requiring higher DA throughput or lower costs, the RDK supports Celestia as an external DA backend, accessed via an IBC connection. The Celestia commitment includes a data root and namespace:

$$\text{CelestiaCommit}(\text{blob}) = (\text{DataRoot}_{\text{Celestia}}, \text{Namespace}, \text{ShareCommitments}, h_{\text{Celestia}}) \tag{7.22}$$

Integration with Celestia occurs through a dedicated IBC relayer module that maintains a verified light client of Celestia's consensus chain. The rollup's data availability verification contract on QoreChain maintains the Celestia light client state, allowing efficient proof verification without requiring full Celestia node infrastructure. Upon receiving a Celestia data blob submission, the RDK performs namespace validation to ensure the blob is correctly namespaced and attaches a Merkle proof over the namespace root to the QoreChain state:

$$\text{CelestiaProof}(\text{blob}, \pi) = \text{MerkleProof}\left(\pi,\ \text{Namespace},\ \text{CelestiaRoot}(h_{\text{Celestia}})\right) \quad (7.23)$$

The "both" DA backend mode provides redundancy by posting data to both native storage and Celestia, with the native copy serving as a fallback if the Celestia IBC channel is temporarily unavailable. This hybrid strategy balances cost efficiency with operational resilience. The rollup specifies a fallback threshold: if Celestia availability samples fail to achieve consensus within a configurable window (typically 4 epochs), the native DA copy is used for settlement. The cost trade-off is quantified as:

$$\text{Cost}_{\text{both}}(\text{blob}) = w_1 \cdot \text{Cost}_{\text{native}}(\text{blob}) + w_2 \cdot \text{Cost}_{\text{Celestia}}(\text{blob}) \quad (7.24)$$

where $w_1, w_2 \in [0, 1]$ with $w_1 + w_2 = 1$ are adjustable weights. For high-throughput rollups, setting $w_1 = 0.2$ and $w_2 = 0.8$ typically achieves 60 percent cost savings while maintaining availability guarantees.

7.4.3 DA Sampling Security

Light nodes (see Chapter 8) perform data availability sampling (DAS) on native DA blobs. Each light node samples k random chunks from an erasure-coded blob. For a blob encoded with rate-1/2 Reed-Solomon coding into $2n$ chunks (where n original chunks can reconstruct the full data):

$$\Pr[\text{accept unavailable blob} \mid k \text{ samples}] \leq \left(\frac{1}{2}\right)^k \quad (7.25)$$

This bound holds because if more than 50% of chunks are missing, the blob cannot be reconstructed, and each random sample has at least a 50% chance of hitting a missing chunk. With $k = 30$ samples per light node:

$$\Pr[\text{single node fooled}] \leq 2^{-30} \approx 9.3 \times 10^{-10} \quad (7.26)$$

For a network of N_L light nodes each independently sampling:

$$\Pr[\text{all nodes fooled}] \leq 2^{-30 \cdot N_L} \quad (7.27)$$

With even 10 light nodes, the probability of an unavailable blob being accepted is 2^{-300}, providing overwhelming security.

Data Availability Operations (CLI)

```
# Submit a DA blob (native backend)
qorchaind tx rdk submit-da-blob \
  --rollup-id "qoredex-rollup-01" \
```

```
  --data ./batch_data.bin \
  --backend native \
  --from qor1sequencer...

# Query DA blob status
qorchaind query rdk da-blob \
  --rollup-id "qoredex-rollup-01" \
  --blob-index 42

# Get DA blob commitment for verification
qorchaind query rdk da-commitment \
  --rollup-id "qoredex-rollup-01" \
  --blob-index 42

# Check DA backend health
qorchaind query rdk da-status --backend celestia
```

7.5 Sequencer Modes

The sequencer determines transaction ordering within the rollup. The choice of sequencer mode balances latency, decentralization, and censorship resistance.

7.5.1 Dedicated Sequencer

A dedicated sequencer is a single operator (typically the rollup creator) that orders transactions and produces batches. This provides the lowest latency but introduces a single point of failure and potential censorship:

$$\text{Latency}_{\text{dedicated}} = t_{\text{receive}} + t_{\text{order}} + t_{\text{batch}} \approx 50\text{–}200 \text{ ms} \tag{7.28}$$

The sequencer operates by receiving transactions from the mempool, validating their signatures and account nonces, ordering them according to a priority function (fee, arrival time, or custom rules), and producing a batch that is submitted to QoreChain's L1. The architecture is stateless for ordering decisions; all rollup state reads occur post-ordering:

$$\text{Batch}_i = (\text{Seq}_i,\ (tx_1, \ldots, tx_n),\ h_{\text{parent}},\ \text{timestamp},\ \text{DA backend choice}) \tag{7.29}$$

The sequencer's incentives align with rollup revenue: a configurable percentage of transaction fees are distributed to the sequencer operator, incentivizing prompt batch production and reliability. To mitigate censorship risk, dedicated sequencers must include a forced inclusion mechanism: if a transaction is submitted to the L1 and not included in a rollup batch within W_{force} blocks, it is automatically force-included. This ensures liveness even if the sequencer becomes unavailable or actively censors transactions:

$$\Pr[\text{censored tx included within } W_{\text{force}} \text{ blocks}] = 1 \tag{7.30}$$

The sequencer's uptime impacts overall rollup stability; operators are incentivized to maintain 99.9 percent availability through reputation systems and per-batch confirmation rewards. Rollups may also employ a secondary sequencer (standby mode) that can activate within 1 epoch if the primary sequencer falls offline, providing fault tolerance without requiring consensus.

7.5.2 Shared Sequencer

The shared sequencer mode distributes sequencing responsibility across QoreChain's validator set. A rotating subset of validators acts as the sequencer committee for each rollup:

$$\text{SeqCommittee}(R, h) = \text{Top-}k\,(\text{ValidatorSet}(h), \text{score}(v, R)) \tag{7.31}$$

where the score function incorporates stake weight, geographic proximity to the rollup's user base, and QCAI performance rating. In shared sequencer mode, the committee rotates every epoch: at epoch transition, the top-k validators by combined score are selected to collectively order all transactions for the assigned rollups. Transactions are ordered via a Byzantine Fault Tolerant (BFT) consensus round among the committee members, where at least 2/3 of committee members must sign off on batch ordering:

$$\Pr[\text{ordering censorship}] \leq \Pr[\text{attack controls } > 1/3 \text{ of committee}] \tag{7.32}$$

Shared sequencing provides stronger decentralization and enables atomic cross-rollup transactions when multiple rollups share the same sequencer committee. Multiple rollups can be assigned to the same sequencer committee, allowing them to exchange liquidity and perform atomic swaps without intermediate settlement:

$$\text{AtomicSwap}(R_1, R_2,\ t_1, t_2) = \text{Sign}_{\text{committee}}\,(\text{BatchSeq}_{1:2}(t_1, t_2)) \tag{7.33}$$

where $\text{BatchSeq}_{1:2}$ produces an ordered sequence containing transactions from both rollups such that constraints (e.g., locked collateral in R_1 is unlocked only if R_2 receives its counterpart) are satisfied atomically. The sequencer committee earns a per-rollup commission proportional to network fees, incentivizing high-quality service and fast batch production. Rollups may also configure custom scoring functions to prefer validators with low latency to their geographic region or those with specialized hardware (GPU acceleration for certain proof verification).

7.5.3 Based Sequencer

Based sequencing (L1-sequenced) delegates transaction ordering entirely to the main chain, as described in Section 7.3. This provides the strongest censorship resistance (equivalent to the main chain) at the cost of higher latency:

$$\text{Latency}_{\text{based}} = t_{\text{L1\_}block} + t_{\text{inclusion}} \approx 6\text{–}12 \text{ s} \tag{7.34}$$

The tradeoff across sequencer modes is formalized as:

$$\text{Decentralization}(\delta) \propto \frac{1}{\text{Latency}(\delta)}, \quad \delta \in \{\text{dedicated}, \text{shared}, \text{based}\} \tag{7.35}$$

Configuring Sequencer Mode (CLI)

```
# Update sequencer mode for an existing rollup
qorchaind tx rdk update-rollup \
  --rollup-id "qoredex-rollup-01" \
  --sequencer-mode shared \
  --from qor1creator...

# Query current sequencer committee (shared mode)
qorchaind query rdk sequencer-committee \
  --rollup-id "qoredex-rollup-01"

# Check force-inclusion queue (dedicated mode)
qorchaind query rdk force-inclusion-queue \
  --rollup-id "qoredex-rollup-01"
```

7.6 Cross-Layer Fee Bundling (CLFB)

Cross-Layer Fee Bundling enables users to pay a single fee for operations that span multiple layers. A CLFB transaction t_{bundle} specifies operations on K layers:

$$t_{\text{bundle}} = ((l_1, \text{op}_1), (l_2, \text{op}_2), \ldots, (l_K, \text{op}_K), \ \text{Fee}_{\text{total}}) \tag{7.36}$$

The total fee is distributed proportionally to gas consumed on each layer:

$$\text{Fee}_{l_j} = \text{Fee}_{\text{total}} \cdot \frac{G_{l_j}}{\sum_{i=1}^{K} G_{l_i}} \tag{7.37}$$

where G_{l_j} is the gas consumed by operation op_j on layer l_j. CLFB transactions are atomic: if any operation on any layer fails, all operations across all layers are reverted.

The fee discount for bundled operations incentivizes multi-layer usage:

$$\text{Fee}_{\text{bundle}} = (1 - d_K) \cdot \sum_{j=1}^{K} \text{Fee}_{\text{standalone}}(\text{op}_j), \quad d_K = \min(0.2, \ 0.05 \cdot (K - 1)) \tag{7.38}$$

where d_K is a discount factor that increases with the number of layers involved, capped at 20%.

7.7 Rollup Economics

7.7.1 Rollup Creation

Creating a rollup requires:

- **Minimum stake:** 10,000 QOR escrowed in the RDK module account. Returned upon graceful rollup shutdown.
- **Creation burn:** 1% of the staked amount is burned via the `BurnSourceRollupCreate` burn channel.
- **Maximum rollups:** The network supports up to 100 concurrent active rollups (governance-adjustable).

The effective cost of rollup creation is:

$$\text{Cost}_{\text{create}} = \text{Stake}_{\text{min}} \times r_{\text{burn}} + \text{gas}_{\text{create}} \cdot P_{\text{gas}} \tag{7.39}$$

where $r_{\text{burn}} = 0.01$ (1%). At a 10,000 QOR minimum stake, the burn cost is 100 QOR.

7.7.2 AI-Optimized Configuration

The QCAI reinforcement learning agent provides two optimization services for rollups:

1. **Profile Suggestion:** Given workload characteristics (expected TPS, average transaction size, value distribution, latency requirements), the RL agent suggests the optimal rollup profile and configuration.
2. **Gas Optimization:** The agent dynamically adjusts the rollup's gas configuration to balance revenue, user cost, and utilization.

The profile suggestion function maps workload features to a recommended configuration:

$$\text{SuggestProfile} : \mathbb{R}^d \to \text{RollupConfig} \tag{7.40}$$

where the input features include expected TPS, average transaction value, required finality time, maximum acceptable cost per transaction, and target decentralization level.

AI-Optimized Rollup Configuration (SDK)

```
import { QoreChainClient } from "@qorechain/sdk";

const client = new QoreChainClient({ rpcEndpoint:
    ↪ "https://rpc.qorechain.io" });

// Get AI-suggested rollup profile
const suggestion = await client.rdk.suggestProfile({
  expectedTPS: 500,
  avgTxValueQOR: 100,
  maxFinalitySeconds: 60,
  maxCostPerTxUQOR: 1000,
  decentralizationPriority: "medium",
});
```

```
console.log(`Recommended profile: ${suggestion.profile}`);
console.log(`Settlement: ${suggestion.settlementMode}`);
console.log(`Sequencer: ${suggestion.sequencerMode}`);
console.log(`DA backend: ${suggestion.daBackend}`);
console.log(`Estimated cost/tx: ${suggestion.estimatedCostPerTx}
    ↪ uqor`);
console.log(`Estimated finality:
    ↪ ${suggestion.estimatedFinalitySeconds}s`);
console.log(`Confidence: ${(suggestion.confidence *
    ↪ 100).toFixed(1)}%`);

// Optimize gas config for an existing rollup
const gasOptimization = await client.rdk.optimizeGasConfig({
  rollupId: "qoredex-rollup-01",
  targetUtilization: 0.75,
  maxPriceMultiplier: 2.0,
});
console.log(`New gas limit: ${gasOptimization.gasLimit}`);
console.log(`New base price: ${gasOptimization.basePrice} uqor`);
```

7.8 Hybrid Chain State (HCS)

Hybrid Chain State (HCS) layers combine on-chain settlement with off-chain computation, providing a flexible execution model for workflows that require selective on-chain visibility. HCS layers maintain a dual-state model:

$$S_{\text{HCS}} = (S_{\text{public}}, S_{\text{private}}, \text{Commit}(S_{\text{private}})) \tag{7.41}$$

where S_{public} is stored on-chain and queryable by any participant, S_{private} is maintained off-chain by the HCS operator, and $\text{Commit}(S_{\text{private}})$ is a PQC-signed commitment to the private state anchored on the main chain.

The commitment ensures that the HCS operator cannot retroactively alter the private state without detection:

$$\text{Verify}_{\text{HCS}}(S_{\text{private}}, \text{Commit}) = \left[H_{\text{SHAKE-256}}(S_{\text{private}}) \stackrel{?}{=} \text{Commit.hash}\right] \wedge \text{VerifyMLDSA}(\text{Commit.sig}) \tag{7.42}$$

HCS layers are suited for enterprise use cases requiring data privacy (e.g., supply chain tracking with confidential business data, healthcare record management with selective disclosure, financial settlement with regulatory compliance).

HCS Layer Operations (CLI)

```
# Register an HCS layer
qorchaind tx multilayer register-hcs \
  --name "Supply Chain Tracker" \
  --operator qor1operator... \
```

```
  --anchor-interval 100 \
  --from qor1operator...

# Submit a state anchoring (PQC-signed commitment)
qorchaind tx multilayer anchor-hcs \
  --layer-id "supply-chain-01" \
  --public-state-root "abc123..." \
  --private-state-commitment "def456..." \
  --pqc-signature ./anchor_sig.bin \
  --from qor1operator...

# Query HCS anchor history
qorchaind query multilayer hcs-anchors \
  --layer-id "supply-chain-01" \
  --limit 10
```

Chapter 8

Light Node Network

8.1 Architecture Overview

The QoreChain Light Node Network is a distributed infrastructure layer that enables participation in network operations without the hardware requirements of a full validator. Light nodes contribute to data availability sampling, state proof verification, transaction relay, and network health monitoring, earning a dedicated 3% share of all network fees in return.

Two editions serve different operator profiles:

Edition	Name	Form Factor	Target Operator	Responsibilities
SX	Server eXperience	Daemon + CLI	Infrastructure operators	Full light node duties: DAS, relay, proof serving
UX	User eXperience	Web dashboard + daemon	General users	Reduced duties: DAS, relay, monitoring

Table 8.1: Light node editions

The SX edition runs as a persistent daemon process suitable for servers and cloud deployments, providing the full range of light node services. The UX edition bundles an embedded web dashboard (built on Alpine.js) with a lightweight daemon, enabling browser-based participation and monitoring. Both editions include a PQC keyring using ML-DSA-87 (Dilithium-5), with Argon2id key derivation and AES-256-GCM encrypted local storage.

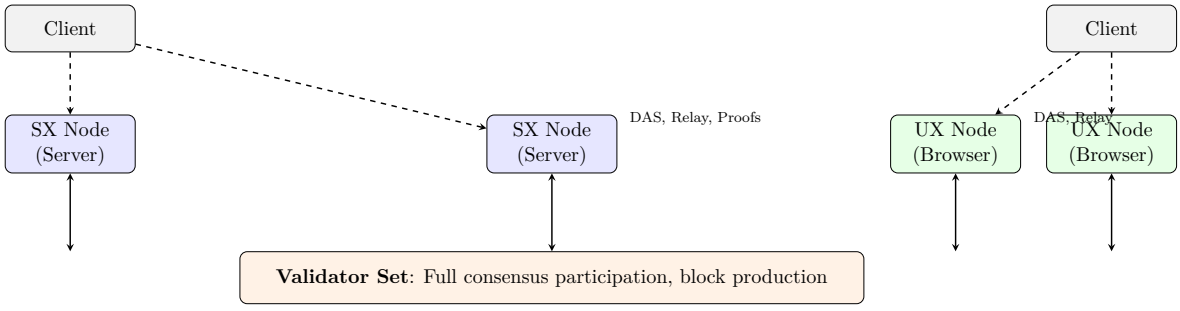

Figure 8.1: Light node network topology

8.2 Registration and Lifecycle

8.2.1 On-Chain Registration

Light node registration is managed by the on-chain `x/lightnode` module. Registration requires a fee of 1 QOR and a minimum delegated stake of 100 QOR. The registration message specifies the node type (SX or UX) and the operator's PQC public key:

$$\text{Register}(v) = (\text{operator}, \text{nodeType} \in \{\text{sx}, \text{ux}\}, \text{pqcPubKey}, \text{version}) \tag{8.1}$$

Upon registration, the node enters a pending state and must submit a proof of connectivity within 72 hours (12,000 blocks). Proof of connectivity is demonstrated by successfully serving a proof sample (DAS challenge) to a subset of other light nodes. Once proof is validated, the node transitions to active status and begins earning rewards proportional to its contribution (DAS samples served, proofs verified, relay traffic handled):

$$\text{Reward}(v, \text{epoch}) = \text{BaseReward} \times \frac{\text{Contribution}(v, \text{epoch})}{\sum_v \text{Contribution}(v, \text{epoch})} \tag{8.2}$$

The network supports a maximum of 10,000 concurrent light nodes (governance-adjustable). To prevent Sybil attacks, each operator may register a maximum of 100 light nodes. An operator's minimum delegated stake increases superlinearly with the number of nodes registered:

$$\text{MinStake}(n_{\text{nodes}}) = 100 \text{ QOR} \times \lceil \sqrt{n_{\text{nodes}}} \rceil \tag{8.3}$$

The registration process is transparent; a registered node's address, stake, node type, and performance metrics are published on-chain. Governance may adjust registration parameters (fee, minimum stake, maximum concurrent nodes, proof-of-connectivity timeout) through standard governance proposals, ensuring the light node network remains cost-effective while maintaining security bounds.

8.2.2 Heartbeat Liveness Protocol

Active light nodes must submit periodic heartbeat transactions to prove liveness. The heartbeat interval I_h is 1,000 blocks (approximately 100 minutes at 6-second block times), with a grace period G_h of 200 blocks:

$$\text{HeartbeatRequired}(v, h) = \begin{cases} \text{true} & \text{if } h - h_{\text{last}}(v) > I_h \\ \text{false} & \text{otherwise} \end{cases} \tag{8.4}$$

A node that misses the heartbeat deadline plus grace period is transitioned to inactive status:

$$\text{Status}(v, h) = \begin{cases} \text{active} & \text{if } h - h_{\text{last}}(v) \leq I_h + G_h \\ \text{inactive} & \text{if } h - h_{\text{last}}(v) > I_h + G_h \end{cases} \tag{8.5}$$

Inactive nodes stop earning rewards but can be re-activated by submitting a heartbeat transaction, which resets $h_{\text{last}}(v)$ to the current block height.

The EndBlocker at each block boundary checks heartbeat grace periods and triggers reward distribution for active nodes.

Light Node Registration and Lifecycle (CLI)

```
# Register as an SX light node
qorchaind tx lightnode register-node \
  --node-type sx \
  --from qor1operator... \
  --fees 1000000uqor

# Send a heartbeat (must be done every ~1000 blocks)
qorchaind tx lightnode heartbeat \
  --from qor1operator...

# Check your light node status
qorchaind query lightnode light-node qor1operator...

# List all active light nodes
qorchaind query lightnode light-nodes --status active

# Deregister (returns delegated stake)
qorchaind tx lightnode deregister-node \
  --from qor1operator...
```

8.3 Light Node Responsibilities

8.3.1 Data Availability Sampling (DAS)

Light nodes perform data availability sampling on new blocks and rollup DA blobs. Each light node independently samples k random chunks from each erasure-coded data object to verify that the data is available for reconstruction.

Erasure Coding. Block data of size D bytes is encoded using rate-1/2 Reed-Solomon coding into $2n$ chunks of size D/n each, where any n chunks suffice to reconstruct the original data:

$$\text{Encode} : \mathbb{F}_q^n \to \mathbb{F}_q^{2n}, \quad \text{Decode} : \text{any } n \text{ of } 2n \text{ chunks} \to \mathbb{F}_q^n \tag{8.6}$$

where $\mathbb{F}_q$ is a finite field of order q (typically $q = 2^{256}$ for compatibility with SHAKE-256 hashes).

Sampling Protocol. Each light node uniformly samples k chunk indices from $\{1, 2, \ldots, 2n\}$ and requests those chunks from the network. A chunk response includes the chunk data and a Merkle proof linking it to the block's data root:

$$\text{Sample}(v) = \left\{(i_j, c_{i_j}, \pi_{i_j})\right\}_{j=1}^{k}, \quad i_j \sim \text{Uniform}(\{1, \ldots, 2n\}) \tag{8.7}$$

where c_{i_j} is the chunk data and π_{i_j} is the Merkle proof of size $O(\log(2n))$.

Acceptance Rule. A light node accepts a block's data as available if and only if all k sampled chunks are returned with valid Merkle proofs:

$$\text{Accept}(v) = \bigwedge_{j=1}^{k} \Big[\text{received}(c_{i_j}) \wedge \text{VerifyMerkle}(\pi_{i_j}, c_{i_j}, \text{dataRoot})\Big] \tag{8.8}$$

Theorem 5 (DAS Security)**.** *If a block's data has fewer than n of $2n$ chunks available (i.e., the data is unrecoverable), then each light node sampling k chunks rejects the block with probability at least $1 - 2^{-k}$. For a network of N_L independent light nodes:*

$$\Pr[\textit{unrecoverable block accepted by all}] \leq 2^{-k \cdot N_L} \tag{8.9}$$

Proof. If fewer than n of $2n$ chunks are available, then at least $n+1$ chunks are missing. The probability that a single uniformly random sample hits an available chunk is at most $\frac{n-1}{2n} < \frac{1}{2}$. Thus, the probability that all k samples hit available chunks is at most $\left(\frac{1}{2}\right)^k = 2^{-k}$. Since light nodes sample independently, the probability that all N_L nodes accept is at most $\left(2^{-k}\right)^{N_L} = 2^{-k \cdot N_L}$. □ □

With the default parameters $k = 30$:

Light Nodes (N_L)	**Failure Probability**	**Security Level**
1	$2^{-30} \approx 9.3 \times 10^{-10}$	30-bit
10	2^{-300}	300-bit
100	2^{-3000}	3000-bit
1,000	2^{-30000}	30000-bit

Table 8.2: DAS security levels by network size ($k = 30$ samples per node)

Even a small light node network provides overwhelming data availability guarantees, far exceeding the 128-bit security standard.

8.3.2 State Proof Verification and Serving

SX light nodes maintain a cache of recent Merkle state proofs and serve them to clients on request. When a client queries a state value v at key p, the light node returns:

$$\text{Response}(p) = (v, \pi_p, h_{\text{proof}}) \tag{8.10}$$

where π_p is the SHAKE-256 Merkle proof of size $O(\log N)$ (with N total state entries) and h_{proof} is the block height at which the proof was generated. The client verifies:

$$\text{VerifyStateProof}(v, \pi_p, \text{stateRoot}(h_{\text{proof}})) \stackrel{?}{=} \text{true} \tag{8.11}$$

Light nodes do not need to store the full state tree; they request proofs from full nodes and cache the results. The cache eviction policy uses a least-recently-used (LRU) strategy with a configurable maximum cache size.

8.3.3 Transaction Relay

Light nodes participate in mempool propagation by relaying transactions between clients and full nodes. The relay protocol implements:

1. **PQC Signature Pre-Validation:** Light nodes verify ML-DSA-87 transaction signatures before forwarding, filtering invalid transactions and reducing load on validators.
2. **Deduplication:** A Bloom filter tracks recently seen transaction hashes to prevent redundant relay.
3. **Priority Routing:** Transactions are forwarded with priority tags aligned with the 5-lane prioritization system (PQC: 100, MEV: 90, AI: 80, Default: 50, Free: 10).

The expected number of relay hops from a client to a validator is:

$$\mathbb{E}[\text{hops}] = 1 + \frac{N_L}{N_L + N_V} \tag{8.12}$$

where N_V is the number of validators. In practice, most transactions reach a validator within 2 hops.

8.3.4 Network Health Monitoring

Light nodes collect telemetry data and report it to the on-chain statistics module:

- **Latency measurements:** Round-trip time to connected peers, tracked in SQLite (9 tables).
- **Block propagation timing:** Time from block production to local receipt.
- **Peer connectivity:** Number and geographic distribution of connected peers.
- **DA sampling success rate:** Fraction of DAS queries that receive valid responses.

The monitoring framework computes per-epoch network health metrics that aggregate light node observations:

$$H_{\text{network}}(e) = \left(\bar{\lambda}(e), \bar{t}_{\text{prop}}(e), P_{\text{active}}(e), S_{\text{DAS}}(e)\right) \tag{8.13}$$

where $\bar{\lambda}(e)$ is the median observed latency, $\bar{t}_{\text{prop}}(e)$ is the mean block propagation time, $P_{\text{active}}(e)$ is the fraction of observed peers that are responsive, and $S_{\text{DAS}}(e)$ is the cumulative DAS success rate. Alerting thresholds trigger automatic notifications when any metric exceeds configured bounds; for instance, $P_{\text{active}} < 0.7$ indicates potential partition, while $\bar{t}_{\text{prop}} > 150\%$ of target indicates congestion or validator performance degradation.

Light nodes additionally track validator uptime by observing block proposals and validating signatures. The on-chain statistics module maintains a rolling 256-block window of per-validator performance, enabling the consensus engine to adjust pool weights and detect Byzantine behaviour. These metrics feed into the QCAI network monitoring system (Chapter 5), enabling early detection of network partitions, performance degradation, and potential attacks.

8.4 Reward Economics

8.4.1 Fee Distribution

Light nodes receive 3% of all network fees, allocated from the fee distribution schedule:

Recipient	Share
Validators	37%
Burned	30%
Treasury	20%
Stakers	10%
Light Nodes	3%

Table 8.3: Network fee distribution

The 3% light node pool is distributed at each EndBlocker cycle among active light nodes proportional to their weighted contribution.

8.4.2 Reward Distribution Function

For an epoch ending at block h, the total light node reward pool is:

$$R_{\text{pool}}(h) = 0.03 \times \text{TotalFees}(h_{\text{prev}}, h) \tag{8.14}$$

Each active light node v_i receives a share proportional to its weight $w(v_i)$:

$$R(v_i, h) = R_{\text{pool}}(h) \times \frac{w(v_i)}{\sum_{j \in \mathcal{V}_{\text{active}}} w(v_j)} \tag{8.15}$$

The weight function combines delegated stake and uptime:

$$w(v_i) = \text{DelegatedStake}(v_i) \times \text{UptimeFactor}(v_i) \tag{8.16}$$

where the uptime factor is computed from the heartbeat record:

$$\text{UptimeFactor}(v_i) = \frac{\text{HeartbeatsReceived}(v_i, \text{epoch})}{\text{HeartbeatsExpected}(\text{epoch})} \tag{8.17}$$

Nodes with uptime below 80% (the `MinUptimeForRewards` parameter) receive zero rewards:

$$R(v_i, h) = \begin{cases} R_{\text{pool}}(h) \times \frac{w(v_i)}{\sum_j w(v_j)} & \text{if UptimeFactor}(v_i) \geq 0.80 \\ 0 & \text{otherwise} \end{cases} \tag{8.18}$$

8.4.3 Expected Yield Analysis

The expected annualized yield for a light node operator depends on the total network fees and the number of competing light nodes. Let F_{annual} be the annualized total network fees:

$$\text{APY}(v_i) = \frac{R(v_i, \text{year})}{\text{DelegatedStake}(v_i)} = \frac{0.03 \times F_{\text{annual}} \times \frac{w(v_i)}{\sum_j w(v_j)}}{\text{DelegatedStake}(v_i)} \tag{8.19}$$

For a node with 100% uptime (UptimeFactor = 1), this simplifies to:

$$\text{APY}(v_i) = \frac{0.03 \times F_{\text{annual}}}{\sum_j \text{DelegatedStake}(v_j)} \tag{8.20}$$

This shows that the APY is independent of an individual node's stake when all nodes have equal uptime, as the yield is proportional to the node's share of total delegated stake.

Querying and Claiming Light Node Rewards (CLI)

```
# Query accumulated rewards for your light node
qorchaind query lightnode rewards qor1operator...

# Claim accumulated rewards
qorchaind tx lightnode claim-rewards \
  --from qor1operator...

# Query network-wide light node statistics
qorchaind query lightnode stats

# Query light node parameters
qorchaind query lightnode params
```

Light Node Rewards (SDK)

```
import { QoreChainClient } from "@qorechain/sdk";

const client = new QoreChainClient({ rpcEndpoint:
    ↪ "https://rpc.qorechain.io" });

// Get light node info and rewards
const nodeInfo = await client.lightnode.getNode("qor1operator...");
console.log(`Status: ${nodeInfo.status}`);
console.log(`Type: ${nodeInfo.nodeType}`); // sx or ux
console.log(`Delegated stake: ${nodeInfo.delegatedStake} uqor`);
console.log(`Accumulated rewards: ${nodeInfo.accumulatedRewards}
    ↪ uqor`);
console.log(`Uptime: ${(nodeInfo.uptimeFactor * 100).toFixed(1)}%`);

// Get network-wide stats
const stats = await client.lightnode.getStats();
console.log(`Total light nodes: ${stats.totalNodes}`);
console.log(`Active SX: ${stats.activeSX}, Active UX:
    ↪ ${stats.activeUX}`);
console.log(`Total delegated stake: ${stats.totalDelegatedStake}
```

```
    ↪ uqor`);
console.log(`Reward pool (current epoch): ${stats.currentRewardPool}
    ↪ uqor`);

// Estimate APY at current network conditions
const apy = await client.lightnode.estimateAPY({
  delegatedStake: "100000000uqor", // 100 QOR
  uptimeFactor: 1.0,
});
console.log(`Estimated APY: ${(apy.annualizedRate *
    ↪ 100).toFixed(2)}%`);
```

8.5 Delegation and Staking

Light nodes require a minimum delegated stake of 100 QOR to be eligible for rewards. Token holders can delegate to light nodes similarly to validator delegation, creating a participation pathway that does not require running any infrastructure.

8.5.1 Delegation Mechanics

Any QOR holder can delegate tokens to a registered light node:

$$\text{Delegate}(d, v, a) : \text{delegator } d \text{ stakes } a \text{ QOR to light node } v \tag{8.21}$$

The delegated stake increases the node's weight in the reward distribution function. Rewards are split between the operator and delegators according to a commission rate set by the operator:

$$R_{\text{operator}}(v) = R(v) \times \text{Commission}(v), \quad R_{\text{delegators}}(v) = R(v) \times (1 - \text{Commission}(v)) \tag{8.22}$$

Delegator rewards are distributed proportionally to each delegator's share of the total delegation. The delegation operation itself is recorded on-chain and is immediately effective; a delegator's share in the reward pool updates at the next reward epoch:

$$\text{DelegatorShare}(d, v) = \frac{\text{Stake}(d, v)}{\sum_{d' \in \text{Delegators}(v)} \text{Stake}(d', v)} \tag{8.23}$$

Delegation can be increased or decreased subject to unbonding constraints. A delegator initiating an unbond incurs a 21-block unbonding period (compatible with Cosmos SDK), after which the stake becomes available. During unbonding, the delegated tokens no longer contribute to the light node's weight. Delegation accepts a minimum of 1 QOR, enabling small holders to participate without economies of scale. An operator can set a commission rate between 0 and 100 percent, with governance authority able to cap maximum commissions at network-wide consensus if needed to prevent predatory fees.

8.5.2 Auto-Compound and Rebalance

The light node client (both SX and UX editions) includes delegation automation features:

1. **Auto-Compound:** Automatically claims and re-delegates accumulated rewards at configurable intervals, maximizing compound yield.
2. **Rebalance:** If a delegator has stakes across multiple light nodes, the rebalancer periodically redistributes to optimize for uptime-weighted yield, moving stake from underperforming nodes to those with higher uptime.

The compound growth with auto-compounding at interval T_c over a period T is:

$$\text{Balance}(T) = S_0 \times \left(1 + \frac{r}{T/T_c}\right)^{T/T_c} \tag{8.24}$$

where S_0 is the initial stake and r is the annual reward rate. As $T_c \to 0$ (continuous compounding):

$$\text{Balance}(T) \to S_0 \times e^{r \cdot T} \tag{8.25}$$

8.6 Registration Model

Light node participation is designed to be accessible. Unlike validators and bridge operators, which require on-chain licenses through the `x/license` module, light nodes register directly through the `x/lightnode` module with minimal requirements:

- **Registration fee:** 1 QOR (one-time, covers on-chain registration transaction).
- **Minimum delegated stake:** 100 QOR (must be maintained for reward eligibility).
- **Heartbeat liveness:** Periodic heartbeat transactions prove continued operation.

Registration is available through the QoreChain dashboard or via CLI. The dashboard provides a guided registration flow: key generation, stake delegation, and automatic heartbeat scheduling.

The heartbeat acceptance condition is:

$$\text{AcceptHeartbeat}(v) = \text{ActiveStatus}(v) \wedge (h - h_{\text{last}}(v) \geq I_h) \tag{8.26}$$

This low barrier to entry is intentional: the light node network is designed as a broad participation layer, while the license system (Chapter 9) gates higher-trust roles such as cross-chain bridge watching and external chain validation.

8.7 UX Edition: Browser-Based Participation

The UX edition lowers the barrier to light node participation by providing a browser-based interface. The embedded web dashboard (built with Alpine.js) provides real-time monitoring and management across 7 pages:

1. **Dashboard:** Network overview, node status, real-time reward tracking via WebSocket.
2. **Delegation:** Manage delegated stake, view delegation history, configure auto-compound.
3. **Rewards:** Accumulated rewards, claim history, APY projections.
4. **Validators:** Browse validator set, view performance metrics, delegate to validators.
5. **Network:** Peer connectivity, block propagation statistics, DAS success rates.
6. **Keys:** PQC key management (generate, backup, restore) with hardware security module support.
7. **Settings:** Node configuration, heartbeat interval, relay preferences.

The UX edition performs a subset of SX duties. Specifically, UX nodes participate in DAS and transaction relay but do not serve full state proofs (which require persistent storage beyond browser capabilities). The reduced duty set is reflected in a lower expected reward weight, incentivizing dedicated SX deployment for operators seeking maximum yield.

Responsibility	SX	UX
Data Availability Sampling	Yes	Yes
Transaction Relay	Yes	Yes
Network Health Monitoring	Yes	Yes
State Proof Serving	Yes	No
Full Merkle Proof Cache	Yes	No
Persistent Storage (SQLite)	Yes	Limited

Table 8.4: Responsibility comparison between SX and UX editions

Running a Light Node (SX and UX)

```
# --- SX Edition (Server) ---

# Initialize and generate PQC keys
qorechain-lightnode init --edition sx --data-dir
    ↪ /var/qorechain/lightnode
qorechain-lightnode keys generate --algorithm dilithium5

# Start the SX daemon
```

```
qorechain-lightnode start \
  --rpc-endpoint https://rpc.qorechain.io \
  --heartbeat-auto \
  --das-samples 30 \
  --relay-enabled \
  --proof-cache-size 1GB

# Check SX status
qorechain-lightnode status

# --- UX Edition (Browser Dashboard) ---

# Initialize UX edition (starts embedded web server)
qorechain-lightnode init --edition ux --data-dir
    ↪ ~/.qorechain-lightnode
qorechain-lightnode keys generate --algorithm dilithium5

# Start UX daemon with web dashboard
qorechain-lightnode start \
  --rpc-endpoint https://rpc.qorechain.io \
  --dashboard-port 8080 \
  --heartbeat-auto

# Open dashboard in browser
# Navigate to http://localhost:8080

# --- Docker (both editions) ---

# SX via Docker
docker run -d --name qorechain-lightnode-sx \
  -v lightnode-data:/data \
  -e RPC_ENDPOINT=https://rpc.qorechain.io \
  qorechain/lightnode:latest --edition sx

# UX via Docker (exposes dashboard on port 8080)
docker run -d --name qorechain-lightnode-ux \
  -v lightnode-data:/data \
  -p 8080:8080 \
  -e RPC_ENDPOINT=https://rpc.qorechain.io \
  qorechain/lightnode:latest --edition ux
```

8.8 Security Considerations

8.8.1 Anti-Sybil Protection

The combination of registration fees, minimum delegated stake, and heartbeat liveness requirements provides multi-layered Sybil resistance:

$$\text{CostPerSybil} = \text{RegistrationFee} + \text{MinDelegatedStake} = 1 + 100 = 101 \text{ QOR} \quad (8.27)$$

An attacker attempting to control a significant fraction of the light node network would need to lock $101 \times N_{\text{sybil}}$ QOR, where N_{sybil} is the number of Sybil nodes. With 10,000 maximum light nodes:

$$\text{Cost To Control } 50\% = 101 \times 5{,}000 = 505{,}000 \text{ QOR} \quad (8.28)$$

Additionally, the heartbeat liveness requirement imposes an ongoing operational cost (transaction fees for periodic heartbeats), making large-scale Sybil attacks economically unattractive.

The cumulative cost barrier grows non-linearly with the attacker's desired control fraction. If heartbeat fees average 0.1 uqor per heartbeat and each light node must submit one heartbeat every 10,000 blocks, the periodic operational cost per Sybil node is:

$$C_{\text{heartbeat}}(N_{\text{blocks}}) = 0.1 \times \frac{N_{\text{blocks}}}{10{,}000} \text{ uqor} \approx 0.038 \text{ QOR per epoch} \quad (8.29)$$

An attacker controlling 5,000 Sybil nodes faces an amortised capital cost of 505,000 QOR (not including opportunity cost) plus ongoing operational costs of approximately 190 QOR per epoch, rendering the attack economically infeasible compared to honest participation or alternative attack vectors. Furthermore, Sybil light nodes contribute reduced value to the network; they perform the same computational duties (DAS sampling, block propagation monitoring) as honest nodes but carry no reputation signal and earn only baseline rewards. This contrasts with attacks on validator networks, where Sybil validators earn proportional stake-based rewards.

8.8.2 Data Integrity

Light nodes do not have write access to the blockchain state. All data served by light nodes (state proofs, block headers, transaction data) is accompanied by cryptographic proofs verifiable against the main chain's state root. A malicious light node cannot serve falsified data without the client detecting the forgery:

$$\Pr[\text{accept forged proof}] \leq 2^{-\lambda} \quad (8.30)$$

where λ is the security parameter of the SHAKE-256 hash function (256-bit security level). Verification of served data follows a commit-and-prove protocol: a light node first commits to a state root hash (obtained from a recent QoreChain block header), then provides Merkle proofs against that root for requested data. Clients independently verify the root against the main chain's canonical state, ensuring the light node cannot serve stale or forged data without detection:

$$\text{Verify}(d, \pi, R) = [\text{MerkleProof}(\pi, d, R) \text{ is valid}] \wedge [\text{Root}(R) \text{ is on-chain recent}] \quad (8.31)$$

For state proofs exceeding a certain size (e.g., batch transaction lists), light nodes can serve proofs in compressed form using Merkle-Patricia trie compressions, reducing bandwidth while maintaining non-interactive verification. Historical data (blocks older than 2 epochs) can be served with additional proofs linking to finalized block headers, ensuring that light nodes cannot retrofit their answers.

8.8.3 Eclipse Attack Resistance

To prevent eclipse attacks (where a client is surrounded by malicious light nodes), the client protocol requires connecting to a minimum of m light nodes and accepting data only when $\lceil m/2 \rceil + 1$ nodes agree:

$$\Pr[\text{eclipse success}] \leq \binom{m}{\lceil m/2 \rceil + 1} \cdot \left(\frac{f}{N_L} \right)^{\lceil m/2 \rceil + 1} \tag{8.32}$$

where f is the number of adversarial light nodes. With $m = 8$ connections and $f/N_L < 0.33$, this probability is negligible. An eclipse attack proceeds through two stages: (1) the attacker controls the client's network connections, isolating it from honest peers, and (2) the attacker serves falsified data agreed upon by its own malicious nodes. The quorum requirement defends against both: even with 8 connections, an attacker controlling fewer than 6 nodes cannot create a consensus. The light node network's geographic diversity further increases attack cost; registering light nodes in different jurisdictions and data centers, enforced through reputation mechanisms and latency-based scoring, makes it difficult for an attacker to occupy the client's peer slots. Additionally, clients employ randomized peer selection: every 24 hours, peers are reshuffled by randomly sampling from the set of registered light nodes with active heartbeats. This prevents an attacker from monopolizing a client's connection set over extended periods, even with moderate resources.

Chapter 9

QoreChain Bridge and Cross-Chain Interoperability

Cross-chain interoperability is a foundational design requirement of the QoreChain architecture. With 25 direct Layer 1 connections spanning every major blockchain family, QoreChain occupies a unique topological position: it serves as a universal routing hub through which any connected chain can reach any other chain in a single hop. This chapter formalises the *One-Hop-Swap* principle, the dual-protocol bridge architecture, the role of QCAI in intelligent route selection and bridge verification, and the layered security model that combines AI-driven anomaly detection with post-quantum cryptographic attestations.

9.1 The One-Hop-Swap Principle

9.1.1 Topological Motivation

Traditional cross-chain transfers between non-adjacent blockchains require traversing multiple intermediate networks. A user wishing to move value from Chain A to Chain B may need to route through chains $X_1, X_2, \ldots, X_k$, each introducing an additional fee event, an additional trust assumption, and an additional latency contribution. Formally, consider a graph $G = (V, E)$ where each vertex $v \in V$ represents a blockchain and each edge $e \in E$ represents a direct bridge connection. The cost of a path $\pi = (v_0, v_1, \ldots, v_k)$ is:

$$C(\pi) = \sum_{i=0}^{k-1} \Big[\phi(v_i, v_{i+1}) + \lambda(v_i, v_{i+1}) + \rho(v_i, v_{i+1}) \Big] \tag{9.1}$$

where ϕ denotes the fee, λ the latency, and ρ the risk premium for each hop. Each additional hop compounds these costs multiplicatively in the worst case, since a failure at any intermediate step can strand assets:

$$\Pr[\text{success}(\pi)] = \prod_{i=0}^{k-1} \Big(1 - p_{\text{fail}}(v_i, v_{i+1}) \Big) \tag{9.2}$$

where $p_{\text{fail}}(v_i, v_{i+1})$ is the probability of failure on each individual hop.

Network topology analysis reveals that cross-chain routing follows a power-law distribution in traditional multi-chain ecosystems; a small number of hub chains (e.g.,

Ethereum) receive most inter-chain traffic, while chains connected only indirectly accumulate high latency. The routing distance $d_G(u, v)$ between chains u and v in an arbitrary bridge graph follows:

$$\mathbb{E}[d_G(u, v)] \approx \frac{\ln |V|}{\ln d_{\text{avg}}} + O(1) \tag{9.3}$$

where d_{avg} is the average degree. For existing bridge ecosystems with limited hub capacity, typical transfer paths require 2 to 5 hops, multiplying fees and failure risk substantially. QoreChain's design targets diameter reduction, ensuring that any connected chain reaches any other in exactly 2 hops through QoreChain, compared to the historical average of 3 to 5 hops in fragmented multi-chain networks.

9.1.2 QoreChain as Universal Hub

QoreChain connects to $d = 25$ Layer 1 blockchains directly. Let $\mathcal{Q}$ denote the QoreChain vertex in G. For any pair of connected chains (A, B) where both A and B share an edge with $\mathcal{Q}$, the transfer path is:

$$\pi_{\text{OHS}} = (A, \mathcal{Q}, B) \tag{9.4}$$

with exactly one intermediate hop. The hop count reduction relative to a general multi-hop path is:

$$\Delta H = |\pi_{\text{multi}}| - |\pi_{\text{OHS}}| = (k + 1) - 2 = k - 1 \tag{9.5}$$

The cost reduction follows directly:

$$C(\pi_{\text{OHS}}) = \phi(A, \mathcal{Q}) + \lambda(A, \mathcal{Q}) + \rho(A, \mathcal{Q}) + \phi(\mathcal{Q}, B) + \lambda(\mathcal{Q}, B) + \rho(\mathcal{Q}, B) \tag{9.6}$$

Theorem 6 (One-Hop Cost Bound)**.** *For any pair of chains (A, B) both connected to QoreChain, and any alternative multi-hop path π' of length $k \geq 3$, the One-Hop-Swap path satisfies:*

$$C(\pi_{OHS}) \leq \frac{2}{k} \cdot C(\pi') + \epsilon$$

where ϵ accounts for the difference in per-hop costs between QoreChain bridge endpoints and the intermediate chains in π'. When per-hop costs are approximately uniform, $\epsilon \to 0$ and the One-Hop-Swap achieves a cost reduction factor of $k/2$.

The success probability improves correspondingly:

$$\Pr[\text{success}(\pi_{\text{OHS}})] = (1 - p_{\text{fail}}(A, \mathcal{Q})) \cdot (1 - p_{\text{fail}}(\mathcal{Q}, B)) \geq \Pr[\text{success}(\pi')] \tag{9.7}$$

for any π' with $|\pi'| > 2$, assuming non-zero per-hop failure probabilities.

9.1.3 Reachability and Connectivity

The connectivity matrix $\mathbf{R} \in \{0,1\}^{d \times d}$ is defined as:

$$R_{ij} = \begin{cases} 1 & \text{if both chain } i \text{ and chain } j \text{ are connected to } \mathcal{Q} \\ 0 & \text{otherwise} \end{cases} \tag{9.8}$$

With $d = 25$ direct connections, the number of unique One-Hop-Swap pairs is:

$$|\{(i,j) : R_{ij} = 1,\ i \neq j\}| = \binom{25}{2} = 300 \tag{9.9}$$

Every pair of connected chains can execute a cross-chain transfer through QoreChain without relying on any third-party intermediate network.

```
# One-Hop-Swap: Transfer USDC from Ethereum to Solana via QoreChain
# Step 1: Bridge USDC from Ethereum to QoreChain
qorechaind tx bridge transfer \
  --from qor1abc...def \
  --source-chain ethereum \
  --amount 1000000000 \
  --denom usdc \
  --chain-id qorechain-diana

# Step 2: Bridge USDC from QoreChain to Solana
qorechaind tx bridge transfer \
  --from qor1abc...def \
  --dest-chain solana \
  --dest-address 7xKXtg2Cw...FkR \
  --amount 1000000000 \
  --denom usdc \
  --chain-id qorechain-diana

# Or use the atomic one-hop swap (single transaction):
qorechaind tx bridge one-hop-swap \
  --from qor1abc...def \
  --source-chain ethereum \
  --dest-chain solana \
  --dest-address 7xKXtg2Cw...FkR \
  --amount 1000000000 \
  --denom usdc \
  --optimize-for fee \
  --chain-id qorechain-diana
```

9.2 Dual-Protocol Architecture: IBC and QCB

QoreChain employs two complementary bridging protocols, each optimised for a different segment of the blockchain ecosystem. Critically, chains in the Cosmos ecosystem are reachable through *both* protocols, enabling QCAI to select the optimal path on a per-transaction basis.

9.2.1 IBC: Inter-Blockchain Communication

QoreChain maintains 8 pre-configured IBC channels to Cosmos-ecosystem chains, operated through the Hermes relayer with automatic client updates, misbehaviour detection, and packet clearing every 100 blocks.

Table 9.1: IBC Channel Configuration

Chain	Prefix	Fee Denom	Integration
Cosmos Hub	`cosmos`	uatom	Core IBC hub
Osmosis	`osmo`	uosmo	DEX liquidity routing
Noble	`noble`	uusdc	Native USDC (gas abstraction)
Celestia	`celestia`	utia	Data availability layer
Stride	`stride`	ustrd	Liquid staking
Akash	`akash`	uakt	Decentralised compute
Babylon	`bbn`	ubbn	BTC restaking checkpoints
Loopback	`qor`	uqor	Internal relay testing

The IBC packet verification follows the standard Cosmos IBC protocol with Merkle proof validation. For a packet p sent from chain S to chain D, the verification function is:

$$\text{Verify}_{\text{IBC}}(p) = \text{MerkleProof}(p.\text{commitment},\ \text{ConsensusState}_S(h)) \wedge h \leq h_{\text{current}} - \Delta h_{\text{trust}} \tag{9.10}$$

where h is the height at which the packet was committed and Δh_{trust} is the trusting period in blocks. The IBC light client tracks the consensus state of the counterparty chain and rejects packets whose proofs reference expired or untrusted heights.

```
# IBC transfer from QoreChain to Osmosis
qorechaind tx ibc-transfer transfer \
  transfer channel-1 \
  osmo1xyz...abc \
  1000000uqor \
  --from qor1abc...def \
  --packet-timeout-height 0-0 \
  --packet-timeout-timestamp 600000000000 \
  --chain-id qorechain-diana

# Query IBC channel status
qorechaind query ibc channel end transfer channel-1

# Query pending IBC packets
qorechaind query ibc channel packet-commitments transfer channel-1
```

9.2.2 QCB: QoreChain Bridge Protocol

The QoreChain Bridge (QCB) is a proprietary bridging protocol designed for non-IBC chains and, importantly, also available as an alternative route for Cosmos-ecosystem

chains. QCB operates through license-gated bridge validators who run chain-specific watcher services and produce PQC-signed attestations for cross-chain events.

The QCB protocol is architected around three core principles. First, it achieves chain-agnostic operation by abstracting the underlying consensus and state proof mechanisms of each connected chain into a standardised event observation and attestation format. Second, it enforces quantum-safe security through mandatory ML-DSA-87 signatures on all attestations, protecting bridge integrity against both current and future adversaries with quantum computing capabilities. Third, it provides operator flexibility through granular feature-based licensing that allows network governance to control which validators may operate watchers for which chains, creating an economic barrier to bridge manipulation while preserving decentralisation incentives.

The QCB protocol lifecycle for a cross-chain event proceeds as follows: (1) a chain-specific watcher service monitors the external chain for protocol-relevant events (asset locks, token mints, state updates); (2) upon detection, the watcher constructs an event observation that includes the event details, chain state commitment, and inclusion proof; (3) the watcher forwards this observation to the QoreChain validator operator; (4) the validator verifies the observation against the external chain's consensus proof and, if valid, cryptographically attests to it using ML-DSA-87; (5) the attestation is submitted on-chain where it is aggregated with attestations from other licensed validators; (6) once the quorum threshold is reached, the event is finalised and the corresponding action (asset transfer, state update) occurs on QoreChain.

This design separates concerns between consensus validation (performed off-chain by individual validators) and quorum formation (performed on-chain atomically), reducing the latency overhead of cross-chain operations while maintaining security guarantees proportional to the bridge validator set.

QCB supports 17 bridge endpoints across 12 chain types, covering every major blockchain architecture:

Table 9.2: QCB Bridge Endpoints

Chain	**Type**	**Confirmations**	**Supported Tokens**
Ethereum	evm	12	ETH, USDC, USDT, WBTC
BSC	evm	15	BNB, BUSD
Solana	solana	32	SOL, USDC
Avalanche	evm	12	AVAX, USDC
Polygon	evm	128	MATIC, USDC
Arbitrum	evm	12	ETH, ARB, USDC
TON	ton	10	TON
Sui	sui_move	3	SUI
Optimism	evm	10	ETH, USDC, OP
Base	evm	10	ETH, USDC
Aptos	aptos_move	6	APT, USDC
Bitcoin	utxo	6	BTC
NEAR	near	3	NEAR
Cardano	cardano	15	ADA
Polkadot	polkadot	12	DOT
Tezos	tezos	2	XTZ
TRON	tron	20	TRX, USDT

The confirmation depth k_c for each chain c is chosen to ensure that the probability of a chain reorganisation affecting a confirmed transaction is bounded:

$$\Pr[\text{reorg depth} \geq k_c] \leq \left(\frac{q_c}{1 - q_c}\right)^{k_c} \tag{9.11}$$

where q_c is the fraction of adversarial hash power (for PoW chains) or adversarial stake (for PoS chains) on chain c. For Polygon ($k_c = 128$), the higher confirmation depth accounts for the chain's faster block times and shallower finality guarantees.

```
# QCB bridge transfer from QoreChain to Ethereum
qorechaind tx bridge transfer \
  --from qor1abc...def \
  --dest-chain ethereum \
  --dest-address 0x742d35Cc6634C0532925a3b844Bc9e7595f2bD18 \
  --amount 5000000000uqor \
  --chain-id qorechain-diana

# Query bridge endpoint status
qorechaind query bridge endpoint ethereum

# Query bridge transfer status by transaction hash
qorechaind query bridge transfer-status \
  --tx-hash 0xABCD...1234

# List all active bridge endpoints with current confirmation depths
qorechaind query bridge endpoints --output json
```

9.2.3 Dual-Protocol Routing for Cosmos Chains

Cosmos-ecosystem chains connected via IBC are *also* reachable through QCB. This dual-protocol coverage is a deliberate design choice: it ensures that no single protocol failure can isolate a connected chain, and it provides QCAI with multiple routing options to optimise for the user's stated preference.

Let $\mathcal{C}_{\text{dual}} \subseteq V$ denote the set of chains reachable through both IBC and QCB. For any chain $c \in \mathcal{C}_{\text{dual}}$, the available route set is:

$$\Pi(c) = \{\pi_{\text{IBC}}(c),\ \pi_{\text{QCB}}(c)\} \tag{9.12}$$

QCAI evaluates both routes and selects the optimal one (see Section 9.3). This redundancy also provides a natural failover mechanism: if IBC relayer latency spikes or a QCB bridge validator set is temporarily below quorum, the alternative protocol remains available.

$$\text{Avail}(c) = 1 - (1 - \text{Avail}_{\text{IBC}}(c)) \cdot (1 - \text{Avail}_{\text{QCB}}(c)) \tag{9.13}$$

For independent failure modes, dual-protocol availability exceeds either protocol's individual availability.

9.3 QCAI Intelligent Route Selection

9.3.1 Multi-Objective Route Optimisation

For every cross-chain operation, QCAI evaluates all available routes in real time and selects the path that best matches the user's stated optimisation preference. The route selection problem is formulated as a multi-objective optimisation over three dimensions: fee, latency, and security.

Let Π denote the set of all feasible routes for a given transfer. Each route $\pi \in \Pi$ is characterised by a feature vector:

$$\mathbf{f}(\pi) = \Big(\text{Fee}(\pi),\ \text{Latency}(\pi),\ \text{Security}(\pi)\Big) \in \mathbb{R}^3 \tag{9.14}$$

The user specifies a preference mode $\mu \in \{\texttt{fee}, \texttt{speed}, \texttt{security}, \texttt{balanced}\}$, which determines the weight vector $\mathbf{w}_\mu$:

$$\mathbf{w}_\mu = \begin{cases} (0.8,\ 0.1,\ 0.1) & \mu = \texttt{fee} \\ (0.1,\ 0.8,\ 0.1) & \mu = \texttt{speed} \\ (0.1,\ 0.1,\ 0.8) & \mu = \texttt{security} \\ (0.34,\ 0.33,\ 0.33) & \mu = \texttt{balanced} \end{cases} \tag{9.15}$$

The composite cost function is:

$$C_\mu(\pi) = w_f \cdot \hat{\phi}(\pi) + w_s \cdot \hat{\lambda}(\pi) + w_\sigma \cdot \frac{1}{\hat{S}(\pi)} \tag{9.16}$$

where $\hat{\phi}$, $\hat{\lambda}$, and $\hat{S}$ are the normalised fee, latency, and security score respectively. QCAI selects:

$$\pi^* = \arg\min_{\pi \in \Pi} C_\mu(\pi) \tag{9.17}$$

9.3.2 Dynamic Route Evaluation

Route features are not static. QCAI continuously monitors network conditions and updates route evaluations in real time. The fee for a route π through protocol $\alpha \in \{\text{IBC}, \text{QCB}\}$ is estimated as:

$$\hat{\phi}_\alpha(\pi) = \text{gas}_{\text{src}} \cdot p^{\text{src}}_{\text{gas}} + \text{bridge\_fee}_\alpha + \text{gas}_{\text{dst}} \cdot \hat{p}^{\text{dst}}_{\text{gas}} \tag{9.18}$$

where $\hat{p}^{\text{dst}}_{\text{gas}}$ is QCAI's predicted gas price on the destination chain, computed using an exponentially weighted moving average (EWMA) of recent gas prices:

$$\hat{p}^{\text{dst}}_{\text{gas}}(t) = \alpha_g \cdot p^{\text{dst}}_{\text{gas}}(t-1) + (1-\alpha_g) \cdot \hat{p}^{\text{dst}}_{\text{gas}}(t-1), \quad \alpha_g \in (0,1) \tag{9.19}$$

The latency estimate incorporates both protocol-specific overhead and current congestion:

$$\hat{\lambda}_\alpha(\pi) = \tau^{\text{src}}_{\text{confirm}} + \tau^{\alpha}_{\text{bridge}} + \tau^{\text{dst}}_{\text{finality}} + \delta^{\alpha}_{\text{congestion}}(t) \tag{9.20}$$

where $\tau^{\text{src}}_{\text{confirm}}$ is the expected confirmation time on the source chain, $\tau^{\alpha}_{\text{bridge}}$ is the protocol-specific bridge processing time, $\tau^{\text{dst}}_{\text{finality}}$ is the finality time on the destination chain, and $\delta^{\alpha}_{\text{congestion}}(t)$ is the current congestion-induced delay estimated by QCAI.

The security score is a composite of multiple factors:

$$\hat{S}_\alpha(\pi) = \beta_1 \cdot Q_\alpha + \beta_2 \cdot D_\alpha + \beta_3 \cdot A_\alpha + \beta_4 \cdot H_\alpha \tag{9.21}$$

where:

- Q_α = bridge validator quorum health (fraction of licensed validators currently active)
- D_α = confirmation depth ratio ($k_c / k_c^{\max}$, normalised)
- A_α = attestation freshness (time since last valid attestation, inversely scored)
- H_α = historical reliability (fraction of successful transfers over a sliding window)

9.3.3 Bellman Optimality for Sequential Routing

When a transfer involves multiple sequential decisions (e.g., choosing protocol for the first leg and token denomination for the swap), the route selection follows a Bellman equation structure:

$$V^*(s) = \min_{a \in \mathcal{A}(s)} \Big[c(s,a) + \gamma \cdot \mathbb{E}_{s' \sim T(s,a)} \Big[V^*(s') \Big] \Big] \tag{9.22}$$

where s is the current routing state (source chain, current token, remaining amount), a is the routing action (protocol choice, denomination), $c(s,a)$ is the immediate cost, γ is a discount factor accounting for time preference, and $T(s,a)$ is the transition distribution capturing network uncertainty. The discount factor $\gamma \in (0,1)$ downweights future costs; typical values are 0.95 to 0.99, reflecting that settling a transfer 10 seconds faster is worth a small premium. The value function $V^*(s)$ is computed by value iteration:

$$V^{(k+1)}(s) = \min_{a \in \mathcal{A}(s)} \Big[c(s,a) + \gamma \cdot \mathbb{E}_{s' \sim T(s,a)} \Big[V^{(k)}(s') \Big] \Big] \tag{9.23}$$

with convergence guaranteed in finite iterations. This ensures that QCAI selects the routing action minimizing total expected cost (fees plus slippage plus time-discounted latency) across all intermediate steps. In practice, the state space is pruned: QCAI considers only chains directly connected to the source and destination, limiting routing depth to at most 3 hops (source chain, relay chain if needed, destination chain). The policy derived from the optimal value function can be precomputed offline and updated hourly as network conditions change, ensuring fast real-time route selection.

9.3.4 Pareto Frontier Analysis

For users who specify `balanced` mode, QCAI computes the Pareto frontier of non-dominated routes and selects the knee point:

Definition 1 (Pareto Dominance)**.** Route π_1 dominates route π_2 (written $\pi_1 \succ \pi_2$) if and only if:

$$\text{Fee}(\pi_1) \leq \text{Fee}(\pi_2) \ \wedge \ \text{Latency}(\pi_1) \leq \text{Latency}(\pi_2) \ \wedge \ \text{Security}(\pi_1) \geq \text{Security}(\pi_2)$$

with at least one strict inequality.

The Pareto set $\Pi^* = \{\pi \in \Pi : \nexists \pi' \in \Pi,\ \pi' \succ \pi\}$ represents all routes where no single objective can be improved without degrading another. The knee point is identified using the maximum curvature method on the normalised Pareto frontier.

```
# Query QCAI route recommendations for a transfer
qorechaind query bridge route-recommend \
  --source-chain ethereum \
  --dest-chain solana \
  --amount 1000000000 \
  --denom usdc \
  --optimize-for fee \
  --output json

# Example response:
# {
#   "recommended_route": {
#     "protocol": "qcb",
#     "path": ["ethereum", "qorechain", "solana"],
#     "estimated_fee": "2.34 USDC",
#     "estimated_latency": "180s",
#     "security_score": 0.97
#   },
#   "alternatives": [
#     {
#       "protocol": "qcb",
#       "path": ["ethereum", "qorechain", "solana"],
#       "estimated_fee": "3.12 USDC",
#       "estimated_latency": "45s",
#       "security_score": 0.99,
#       "note": "Faster but higher fee"
#     }
#   ]
# }

# Specify user preference explicitly
qorechaind tx bridge transfer \
  --from qor1abc...def \
  --dest-chain solana \
  --dest-address 7xKXtg2Cw...FkR \
  --amount 1000000000 \
  --denom usdc \
  --route-preference speed \
  --chain-id qorechain-diana
```

9.4 AI-Powered Bridge Verification

Every bridge transaction on QoreChain passes through a multi-layer AI verification pipeline before execution. QCAI operates at the protocol level, scoring transactions

for anomalies, fraud indicators, and cross-endpoint correlation patterns. This section formalises each verification layer.

9.4.1 Three-Layer Verification Architecture

The AI bridge verification pipeline consists of three sequential layers, each producing an independent assessment that contributes to a composite verdict:

1. **Layer 1: Statistical Anomaly Detection.** Identifies transactions whose features deviate significantly from the learned distribution of normal bridge activity.
2. **Layer 2: Fraud Scoring.** Applies ensemble classification to produce a confidence-weighted fraud probability for flagged transactions.
3. **Layer 3: Cross-Endpoint Correlation.** Detects coordinated attack patterns across multiple bridge endpoints simultaneously.

The composite verdict $\mathcal{V}$ is computed as:

$$\mathcal{V}(tx) = \begin{cases} \texttt{REJECT} & \text{if } \max(S_1, S_2, S_3) > \theta_{\text{reject}} \\ \texttt{FLAG} & \text{if } w_1 S_1 + w_2 S_2 + w_3 S_3 > \theta_{\text{flag}} \\ \texttt{ALLOW} & \text{otherwise} \end{cases} \tag{9.24}$$

where S_1, S_2, S_3 are the scores from each verification layer, w_1, w_2, w_3 are layer weights, and $\theta_{\text{reject}}, \theta_{\text{flag}}$ are governance-configurable thresholds.

9.4.2 Layer 1: Statistical Anomaly Detection

For each bridge endpoint c, QCAI maintains a running statistical model of normal transaction behaviour. Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be the feature vector of a bridge transaction, where features include transfer amount, gas price, sender activity frequency, destination address novelty, time-of-day encoding, and bridge endpoint congestion level.

The anomaly score uses the Mahalanobis distance from the learned distribution:

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})} \tag{9.25}$$

where $\boldsymbol{\mu}$ is the mean feature vector and $\boldsymbol{\Sigma}$ is the covariance matrix, both updated incrementally using Welford's online algorithm for numerical stability:

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} + \frac{\mathbf{x}_t - \boldsymbol{\mu}_{t-1}}{t} \tag{9.26}$$

$$\mathbf{M}_{2,t} = \mathbf{M}_{2,t-1} + (\mathbf{x}_t - \boldsymbol{\mu}_{t-1}) \odot (\mathbf{x}_t - \boldsymbol{\mu}_t) \tag{9.27}$$

$$\boldsymbol{\Sigma}_t = \frac{\mathbf{M}_{2,t}}{t} \tag{9.28}$$

A transaction is flagged as anomalous when $D_M(\mathbf{x}) > \chi^2_{n,1-\alpha}$, where $\chi^2_{n,1-\alpha}$ is the chi-squared critical value at significance level α with n degrees of freedom. The anomaly score for Layer 1 is:

$$S_1(\mathbf{x}) = 1 - F_{\chi^2_n}(D^2_M(\mathbf{x})) \tag{9.29}$$

where $F_{\chi^2_n}$ is the CDF of the chi-squared distribution with n degrees of freedom. This maps the Mahalanobis distance to a $[0, 1]$ anomaly probability.

9.4.3 Layer 2: Isolation Forest Fraud Scoring

For transactions flagged by Layer 1 (or all transactions during heightened security periods), QCAI applies an isolation forest ensemble to produce a fraud score. The isolation forest operates by recursively partitioning the feature space and measuring how quickly a sample is isolated:

$$s(\mathbf{x}, n) = 2^{-\frac{E[h(\mathbf{x})]}{c(n)}} \tag{9.30}$$

where $h(\mathbf{x})$ is the path length to isolate sample $\mathbf{x}$, $E[h(\mathbf{x})]$ is the average over all trees in the forest, and $c(n)$ is the average path length of unsuccessful search in a binary search tree of n elements:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}, \quad H(k) = \ln(k) + \gamma_E \tag{9.31}$$

where $\gamma_E \approx 0.5772$ is the Euler-Mascheroni constant. Scores near 1.0 indicate strong anomaly (quick isolation); scores near 0.5 indicate normal behaviour.

The Bayesian posterior fraud probability, incorporating the isolation score as a likelihood, is:

$$P(\text{fraud} \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid \text{fraud}) \cdot P(\text{fraud})}{P(\mathbf{x} \mid \text{fraud}) \cdot P(\text{fraud}) + P(\mathbf{x} \mid \text{normal}) \cdot P(\text{normal})} \tag{9.32}$$

where $P(\text{fraud})$ is the prior fraud rate (estimated from historical bridge data) and the likelihoods are derived from the isolation forest score distributions for fraud and normal classes respectively. The Layer 2 score is:

$$S_2(\mathbf{x}) = P(\text{fraud} \mid \mathbf{x}) \tag{9.33}$$

9.4.4 Layer 3: Cross-Endpoint Correlation Analysis

Sophisticated attacks may coordinate activity across multiple bridge endpoints simultaneously (e.g., draining liquidity from Ethereum and BSC bridges in parallel). QCAI detects such patterns by computing the cross-correlation matrix of bridge activity across all $m = 17$ QCB endpoints.

Let $\mathbf{v}(t) = (v_1(t), v_2(t), \ldots, v_m(t))$ be the vector of bridge volumes at time t. The normalised cross-correlation between endpoints i and j over a sliding window of width W is:

$$\rho_{ij}(t) = \frac{\sum_{\tau=t-W}^{t} (v_i(\tau) - \bar{v}_i)(v_j(\tau) - \bar{v}_j)}{\sqrt{\sum_{\tau=t-W}^{t} (v_i(\tau) - \bar{v}_i)^2 \cdot \sum_{\tau=t-W}^{t} (v_j(\tau) - \bar{v}_j)^2}} \tag{9.34}$$

Under normal conditions, bridge endpoint volumes are approximately independent ($\rho_{ij} \approx 0$ for $i \neq j$). A coordinated attack manifests as a sudden spike in cross-correlation. The correlation anomaly score is:

$$S_3(t) = \frac{1}{\binom{m}{2}} \sum_{i<j} \max\Big(0,\ |\rho_{ij}(t)| - \rho_{\text{baseline}}\Big) \tag{9.35}$$

where ρ_{baseline} is the average absolute correlation under normal operations, learned from historical data.

Theorem 7 (Cross-Correlation Detection Bound)**.** *For m independent bridge endpoints with normally distributed volumes, the probability that $S_3(t) > \theta$ under the null hypothesis (no coordinated attack) is bounded by:*

$$\Pr[S_3(t) > \theta \mid H_0] \leq \binom{m}{2} \cdot \exp\left(-\frac{W \cdot \theta^2}{2}\right)$$

This ensures that for $W = 100$ blocks and $\theta = 0.3$, the false alarm rate is below 10^{-6}.

9.4.5 AI Verdict Confidence and ROC Analysis

The composite verdict system is calibrated to maintain a target receiver operating characteristic (ROC). Let $\text{TPR}(\theta)$ and $\text{FPR}(\theta)$ denote the true positive rate and false positive rate as functions of the decision threshold θ. The area under the ROC curve (AUC) quantifies the overall discriminative power:

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}^{-1}(u))\, du \tag{9.36}$$

QCAI targets $\text{AUC} \geq 0.995$ for the composite verdict system, achieved through ensemble combination of the three layers. The optimal threshold θ^* is selected by minimising the Bayes risk:

$$\theta^* = \arg\min_{\theta} \Big[c_{\text{FP}} \cdot \text{FPR}(\theta) + c_{\text{FN}} \cdot (1 - \text{TPR}(\theta))\Big] \tag{9.37}$$

where c_{FP} is the cost of a false positive (legitimate transfer blocked) and c_{FN} is the cost of a false negative (fraudulent transfer permitted). Given the asymmetric nature of bridge security, $c_{\text{FN}} \gg c_{\text{FP}}$.

```
# Query AI verification status for a bridge transaction
qorechaind query ai bridge-verdict \
  --tx-hash 0xABCD...1234 \
  --output json

# Example response:
# {
#   "verdict": "ALLOW",
#   "confidence": 0.98,
#   "layer_scores": {
#     "anomaly_detection": 0.12,
#     "fraud_scoring": 0.05,
#     "cross_correlation": 0.02
#   },
#   "processing_time_ms": 3
# }
```

```
# Query current anomaly thresholds (governance-configurable)
qorechaind query ai bridge-thresholds

# Query cross-endpoint correlation status
qorechaind query ai correlation-matrix --output json
```

9.5 PQC-Secured Bridge Attestations

9.5.1 Quantum Threat Model for Bridges

Cross-chain bridges represent a particularly attractive target for "harvest now, decrypt later" attacks. Bridge attestations contain cryptographic commitments to asset locks and transfers that, if forged, could enable the creation of unbacked tokens on the destination chain. An adversary with access to a future quantum computer could retroactively forge bridge proofs if those proofs rely solely on classical cryptographic assumptions. Specific attack vectors include: (1) forging signatures from bridge validators by executing Shor's algorithm on ECDSA key material, (2) breaking hash collisions in attestation commitments via Grover search, (3) recovering encrypted bridge payloads committed with classical KEM, and (4) breaking time-lock puzzles used in challenge escrows. The harvest-now-decrypt-later timeline is particularly concerning for bridges: an attacker could collect bridge attestations for weeks or months, then decrypt them retroactively to forge withdrawals.

QoreChain mitigates this threat by requiring all bridge attestations to be signed with ML-DSA-87 (Dilithium-5, FIPS 204), and all bridge operation payloads to be committed using ML-KEM-1024 (FIPS 203). Attestations are time-stamped with a cryptographic hash over the current QoreChain block (including its timestamp and previous block hash), making it computationally difficult to forge an attestation that matches a particular historical timestamp. Bridge validators cannot revoke or update old attestations; instead, updates require a new attestation with a later timestamp, creating an immutable audit trail. Additionally, all bridge state transitions are published on the QoreChain main chain within 2 blocks, so even if an attacker forges an attestation, honest validators can detect the discrepancy and initiate a safety challenge.

9.5.2 Attestation Structure

A bridge attestation $\mathcal{A}$ for a cross-chain event e on source chain c has the following structure:

$$\mathcal{A}(e, v) = \Big(e,\ \sigma_{\text{classical}}(e, v),\ \sigma_{\text{PQC}}(e, v),\ \text{pk}_v^{\text{PQC}},\ h_c,\ t\Big) \tag{9.38}$$

where $\sigma_{\text{classical}}$ is the Ed25519 signature by validator v, σ_{PQC} is the ML-DSA-87 signature, pk_v^{PQC} is the validator's registered PQC public key, h_c is the block height on source chain c at which the event was observed, and t is the timestamp.

Hybrid verification requires *both* signatures to be valid:

$$\text{Verify}_{\text{hybrid}}(\mathcal{A}) = \text{Ed25519.Verify}(\sigma_{\text{classical}}, e, \text{pk}_v^{\text{Ed25519}}) \wedge \text{ML-DSA.Verify}(\sigma_{\text{PQC}}, e, \text{pk}_v^{\text{PQC}}) \tag{9.39}$$

This ensures backward compatibility with existing classical infrastructure while providing quantum-safe forward security.

9.5.3 Attestation Aggregation

For a bridge transfer to be accepted, a quorum of licensed bridge validators must produce valid attestations. Let L_c denote the set of validators licensed for chain c, and let $n_c = |L_c|$. The quorum threshold is:

$$Q_c = \left\lceil \frac{2}{3} \cdot n_c \right\rceil + 1 \tag{9.40}$$

The aggregated attestation is valid if and only if at least Q_c individual attestations pass hybrid verification:

$$\text{Verify}_{\text{agg}}(e) = \left| \{ v \in L_c : \text{Verify}_{\text{hybrid}}(\mathcal{A}(e, v)) = \top \} \right| \geq Q_c \tag{9.41}$$

The verification cost scales linearly with the quorum size:

$$T_{\text{verify}} = Q_c \cdot \left(T_{\text{Ed25519}} + T_{\text{ML-DSA-87}} \right) \tag{9.42}$$

where $T_{\text{Ed25519}} \approx 70\,\mu\text{s}$ and $T_{\text{ML-DSA-87}} \approx 330\,\mu\text{s}$ per verification. Even with $Q_c = 20$, total verification completes in under 10 ms.

9.5.4 ML-KEM-1024 Payload Commitments

Bridge operation payloads (asset lock proofs, transfer metadata) are additionally protected with ML-KEM-1024 key encapsulation. The commitment scheme works as follows:

1. The bridge coordinator generates an ephemeral ML-KEM-1024 keypair $(pk_{\text{eph}}, sk_{\text{eph}})$.
2. Each participating validator encapsulates a shared secret:
$$(ct_v, ss_v) \leftarrow \text{ML-KEM.Encaps}(pk_{\text{eph}})$$
3. The payload is encrypted under the combined shared secret:
$$E = \text{SHAKE-256}(ss_1 \| ss_2 \| \cdots \| ss_{Q_c}) \oplus \text{payload}$$
4. The commitment $\text{Com}(E)$ is anchored on-chain.

This ensures that even if a quantum adversary intercepts bridge communications, they cannot reconstruct the payload without breaking ML-KEM-1024, which is designed to resist quantum attacks up to at least NIST Security Level 5 ($\geq 2^{256}$ quantum operations).

9.5.5 Quantum Security Bound

The security of bridge attestations against a quantum adversary is governed by the hardness of the Module Learning With Errors (MLWE) problem underlying ML-DSA-87:

$$\mathrm{Adv}^{\mathrm{forge}}_{\text{ML-DSA-87}}(\mathcal{A}_q) \leq \mathrm{negl}(\lambda) \tag{9.43}$$

for any quantum polynomial-time adversary $\mathcal{A}_q$ and security parameter $\lambda = 256$. The concrete security estimate for ML-DSA-87 is approximately 2^{128} quantum gates for the best known lattice attacks, compared to approximately 2^{64} quantum gates to break ECDSA-256 via Shor's algorithm. This provides at least a 2^{64} security margin:

$$\Delta_{\text{security}} = \log_2 \left(\frac{T^{\text{ML-DSA-87}}_{\text{break}}}{T^{\text{ECDSA-256}}_{\text{break}}} \right) \geq 64 \text{ bits} \tag{9.44}$$

```
# Query PQC attestation for a specific bridge event
qorechaind query bridge attestation \
  --event-id bridge-eth-00042 \
  --output json

# Example response:
# {
#   "event_id": "bridge-eth-00042",
#   "source_chain": "ethereum",
#   "attestations_received": 15,
#   "quorum_required": 12,
#   "quorum_met": true,
#   "verification": {
#     "classical_valid": 15,
#     "pqc_valid": 15,
#     "hybrid_valid": 15
#   },
#   "pqc_algorithm": "ML-DSA-87",
#   "payload_commitment": "SHAKE256:0xABCD..."
# }

# Verify a bridge attestation manually
qorechaind query bridge verify-attestation \
  --event-id bridge-eth-00042 \
  --validator qorvaloper1xyz...abc
```

9.6 License-Gated Validator Bridge

9.6.1 On-Chain License Registry

The `x/license` module provides a granular access-control system that determines which validators may operate bridge watchers and external chain validators. Unlike the light node network (which requires only registration), bridge operations demand elevated trust and are gated by on-chain licenses administered through governance.

The license registry defines 27 feature IDs across two categories:

- **Bridge Watching (17 chains):** Ethereum, BSC, Solana, Avalanche, Polygon, Arbitrum, TON, Sui, Optimism, Base, Aptos, Bitcoin, NEAR, Cardano, Polkadot, Tezos, Tron
- **Validator Operations:** `validator_ethereum`, `validator_solana`, and corresponding IDs for chains where QoreChain validators also participate in external consensus

The `x/license` module stores validator licenses in a key-value structure indexed by (v, f) pairs, where v is the validator address and f is the feature ID. Each license record contains the grant timestamp, TTL, current status, and optional metadata (e.g., key material, attestation history). Governance can grant, renew, suspend, or revoke licenses through multi-signature proposals requiring supermajority approval ($2/3 + 1$ of active validators). License grants are propagated via the consensus layer; all validators learn of new licenses within a single block.

9.6.2 License Lifecycle

Each license follows a governed lifecycle:

$$\text{Grant} \xrightarrow{\text{activate}} \text{Active} \xrightarrow[\text{governance}]{\text{suspend}} \text{Suspended} \xrightarrow{\text{revoke}} \text{Revoked} \tag{9.45}$$

Licenses carry a time-to-live (TTL) parameter. The auto-expiry mechanism ensures stale grants do not persist:

$$\text{Status}(l, t) = \begin{cases} \text{Expired} & \text{if } t > t_{\text{grant}}(l) + \text{TTL}(l) \\ \text{Active} & \text{if } t \leq t_{\text{grant}}(l) + \text{TTL}(l) \ \wedge \ \neg\text{Suspended}(l) \\ \text{Suspended} & \text{if governance action pending} \end{cases} \tag{9.46}$$

9.6.3 Minimum License Coverage

For each bridge endpoint c, a minimum number of licensed validators must be active to ensure liveness. Let $L_c^{\text{active}}(t)$ denote the number of active licensed validators for chain c at time t. The liveness condition is:

$$L_c^{\text{active}}(t) \geq L_c^{\min} = \max\left(3, \ \left\lceil \frac{n_c}{3} \right\rceil\right) \tag{9.47}$$

The minimum is set to the greater of 3 validators (ensuring Byzantine resilience with 1/3 dishonest tolerance) or one-third of the main validator set, ensuring that a supermajority of QoreChain validators (2/3 of the main set) are present to sign any bridge operation. If $L_c^{\text{active}}(t) < L_c^{\min}$, the bridge endpoint for chain c enters a degraded mode: new outbound transfers are paused, but inbound transfers that already have sufficient attestations continue to settle. This prevents liveness blocks while maintaining safety; an inbound transfer that has already received 2/3 attestation quorum is guaranteed to settle even if additional validators go offline.

Governance can dynamically adjust $L_c^{\min}$ for specific chains based on their security posture, adoption, and strategic importance. High-value endpoints (e.g., Ethereum)

may require higher minimum coverage (e.g., 20 percent of the validator set); new chains may temporarily operate with the minimum floor of 3 validators until governance upgrades them. Each licensed validator for chain c earns a fixed per-epoch subsidy from the bridge operations revenue pool, incentivizing comprehensive coverage. If coverage falls below the minimum, the subsidy increases (inverse function of active licenses) to recruit additional validators:

$$\text{Subsidy}(c) = \text{BaseSubsidy} \times \frac{L_c^{\min}}{L_c^{\text{active}}(t) + 1} \tag{9.48}$$

This economic mechanism ensures the network self-heals: high demand for bridge services combined with subsidy incentives automatically attracts new licensed validators until coverage is restored.

9.6.4 Watcher Orchestration

Licensed validators operate chain-specific watcher services through an automated orchestration layer. For each licensed chain, the validator's node manages a containerised watcher that monitors the external chain for bridge-relevant events and reports them to the QoreChain consensus.

The orchestrator handles:

- **Container Lifecycle:** Automated provisioning, startup, graceful shutdown, and restart on failure for each chain-specific watcher.
- **Health Monitoring:** Periodic health checks with configurable thresholds. A watcher that fails k consecutive health checks is restarted automatically.
- **Credential Exchange:** Secure credential provisioning to watcher containers, with automatic rotation aligned to the license TTL.

The expected availability of an auxiliary service, given a mean time between failures (MTBF) of μ_f and a mean time to restart (MTTR) of μ_r, is:

$$A_{\text{aux}} = \frac{\mu_f}{\mu_f + \mu_r} \tag{9.49}$$

With container restart times typically under 10 seconds and MTBF measured in days, auxiliary-service availability exceeds 99.99%.

```
# Query license status for a validator
qorechaind query license validator \
  --validator qorvaloper1xyz...abc \
  --output json

# Example response:
# {
#   "validator": "qorvaloper1xyz...abc",
#   "licenses": [
#     {
#       "feature_id": "bridge_ethereum",
#       "status": "active",
#       "granted_at": "2026-03-01T00:00:00Z",
#       "expires_at": "2027-03-01T00:00:00Z"
#     },
#     {
#       "feature_id": "bridge_solana",
#       "status": "active",
#       "granted_at": "2026-03-15T00:00:00Z",
#       "expires_at": "2027-03-15T00:00:00Z"
#     }
#   ],
#   "watcher_status": {
#     "ethereum": "running",
#     "solana": "running"
#   }
# }

# Query minimum license coverage per chain
qorechaind query license coverage --output json

# Query watcher health for a specific chain
qorechaind query license watcher-health ethereum
```

9.7 Bridge Security: Circuit Breakers and Rate Limiting

9.7.1 Circuit Breaker Mechanism

Each bridge endpoint operates an independent circuit breaker that monitors transfer volume and automatically halts operations when anomalous activity is detected. The circuit breaker uses a sliding window of W blocks and triggers when the observed volume exceeds a dynamic threshold.

Let $V_c(t)$ denote the total bridge volume for endpoint c at block t. The moving average and standard deviation over the sliding window are:

$$\bar{V}_c(t) = \frac{1}{W} \sum_{\tau=t-W+1}^{t} V_c(\tau) \tag{9.50}$$

$$\sigma_{V_c}(t) = \sqrt{\frac{1}{W} \sum_{\tau=t-W+1}^{t} \left(V_c(\tau) - \bar{V}_c(t)\right)^2} \tag{9.51}$$

The circuit breaker triggers when:

$$V_c(t) > \bar{V}_c(t) + \kappa \cdot \sigma_{V_c}(t), \quad \kappa \geq 3 \tag{9.52}$$

where κ is a governance-configurable sensitivity parameter. With $\kappa = 3$, the false trigger rate under normally distributed volumes is approximately 0.13%.

9.7.2 Circuit Breaker State Machine

The circuit breaker for each endpoint follows a three-state machine:

$$\texttt{Normal} \xrightarrow{V_c(t)>\text{threshold}} \texttt{Triggered} \xrightarrow{\text{cooldown expires}} \texttt{Cooldown} \xrightarrow{\text{volume normalises}} \texttt{Normal} \tag{9.53}$$

During the `Triggered` state, all new outbound transfers through endpoint c are paused. Inbound transfers with sufficient attestations continue to settle. The cooldown period T_{cool} is governance-configurable and defaults to 50 blocks.

The expected time in triggered state before recovery, assuming the anomalous volume was transient, is:

$$\mathbb{E}[T_{\text{recovery}}] = T_{\text{cool}} + \frac{1}{\mu_{\text{normalise}}} \tag{9.54}$$

where $\mu_{\text{normalise}}$ is the rate at which volume returns to the normal distribution after the anomalous event.

9.7.3 Per-Endpoint Rate Limiting

Independent of the circuit breaker, each bridge endpoint enforces per-block and per-epoch rate limits:

$$\text{RateLimit}_c = \begin{cases} V_c^{\text{block}} \leq V_c^{\text{max,block}} & \text{(per-block cap)} \\ \sum_{t \in \text{epoch}} V_c(t) \leq V_c^{\text{max,epoch}} & \text{(per-epoch cap)} \end{cases} \tag{9.55}$$

These limits are set per chain based on the liquidity depth and risk profile of each endpoint. Transfers exceeding the per-block cap are queued for the next block; transfers that would cause the epoch cap to be exceeded enter a challenge queue with a time-locked release.

9.7.4 Large Withdrawal Challenge Period

Withdrawals exceeding a configurable threshold V_{large} are subject to a challenge period during which the transfer can be disputed:

$$\text{FinaliseWithdrawal}(tx) = \begin{cases} \text{immediate} & \text{if } V(tx) \leq V_{\text{large}} \\ \text{delayed by } T_{\text{challenge}} & \text{if } V(tx) > V_{\text{large}} \ \wedge \ \text{no dispute} \\ \text{cancelled} & \text{if dispute upheld} \end{cases} \tag{9.56}$$

During the challenge period, QCAI performs extended verification: deeper anomaly analysis, destination address reputation checking, and cross-reference with historical fraud patterns.

9.7.5 Confirmation Depth Security

The confirmation depth k_c for each chain is calibrated to bound the reorganisation probability. For a Proof-of-Work chain with adversarial hash rate fraction q_c, the probability of a successful double-spend attack after k_c confirmations follows the Nakamoto bound:

$$P_{\text{attack}}(k_c, q_c) = 1 - \sum_{j=0}^{k_c} \frac{(\lambda k_c)^j e^{-\lambda k_c}}{j!} \left(1 - \left(\frac{q_c}{1-q_c}\right)^{k_c - j}\right) \tag{9.57}$$

where $\lambda = k_c \cdot \frac{q_c}{1-q_c}$. For Proof-of-Stake chains, the confirmation depth accounts for the finality gadget: once a block is finalised, reorganisation requires corrupting more than $\frac{1}{3}$ of the validator set.

Table 9.3: Confirmation Depth Security Analysis

Chain	k_c	**Assumed** q_c	$P_{\textbf{attack}}$
Bitcoin	6	0.25	$< 10^{-3}$
Ethereum	12	0.10	$< 10^{-7}$
Polygon	128	0.20	$< 10^{-15}$
Solana	32	0.10	$< 10^{-12}$
TRON	20	0.15	$< 10^{-8}$

```
# Query circuit breaker status for all endpoints
qorechaind query bridge circuit-breaker --output json

# Example response:
# {
#   "endpoints": [
#     {
#       "chain": "ethereum",
#       "status": "normal",
#       "current_volume": "1234567890",
#       "threshold": "9876543210",
#       "window_blocks": 50
```

```
#     },
#     {
#       "chain": "polygon",
#       "status": "triggered",
#       "triggered_at_block": 142857,
#       "cooldown_remaining": 23,
#       "trigger_reason": "volume_spike"
#     }
#   ]
# }

# Query rate limits for a specific endpoint
qorechaind query bridge rate-limit ethereum

# Query pending large withdrawals in challenge period
qorechaind query bridge pending-challenges --output json
```

9.8 Cross-Chain Fee Bundling and Optimisation

9.8.1 Total Cost of Transfer

The total cost of a cross-chain transfer through QoreChain is decomposed into four components:

$$\text{TCT}(\pi) = \underbrace{\text{gas}_{\text{src}} \cdot p_{\text{gas}}^{\text{src}}}_{\text{source chain fee}} + \underbrace{\phi_{\text{bridge}}(\pi)}_{\text{bridge protocol fee}} + \underbrace{\text{gas}_{\text{dst}} \cdot p_{\text{gas}}^{\text{dst}}}_{\text{destination chain fee}} + \underbrace{\delta_{\text{slippage}}(\pi, V)}_{\text{slippage cost}} \tag{9.58}$$

where V is the transfer volume and δ_{slippage} is the expected slippage for volume V given current liquidity. The bridge protocol fee $\phi_{\text{bridge}}(\pi)$ is determined by the bridge type and route security parameters. For IBC bridges (low latency, high frequency), $\phi_{\text{IBC}} \in [0.001\%, 0.05\%]$ of transfer volume; for QCB bridges (higher security attestation overhead), $\phi_{\text{QCB}} \in [0.05\%, 0.2\%]$. The destination chain gas cost $\text{gas}_{\text{dst}} \cdot p_{\text{gas}}^{\text{dst}}$ is hard to predict in advance; QCAI estimates it using historical data and current mempool congestion. The slippage cost varies with DEX depth on the destination chain and the user's tolerance for acceptable execution price:

$$\delta_{\text{slippage}}(\pi, V) = V \times (\text{ExecutionPrice}(V) - \text{BestPrice}) \tag{9.59}$$

where BestPrice is the price at zero volume. For typical transfers (less than USD 100,000), slippage is negligible; for large transfers (USD 1M+), slippage can dominate total cost. The user specifies a maximum acceptable slippage (e.g., 0.5 percent), and QCAI only recommends routes satisfying that constraint. Total cost is also affected by bridge state: if a bridge endpoint is in degraded mode (insufficient licensed validators), QCAI increases ϕ_{bridge} as a risk premium or suggests an alternative route entirely.

9.8.2 One-Hop Cost Advantage

For a multi-hop path $\pi_{\text{multi}} = (A, X_1, \ldots, X_{k-1}, B)$ with k intermediate hops, the total cost is:

$$\mathrm{TCT}(\pi_{\mathrm{multi}}) = \sum_{i=0}^{k} \Big[\mathrm{gas}_i \cdot p_i + \phi_i + \delta_i(V_i) \Big] \tag{9.60}$$

where V_i is the volume at hop i (decreasing due to accumulated fees). The One-Hop-Swap path has:

$$\mathrm{TCT}(\pi_{\mathrm{OHS}}) = \mathrm{gas}_A \cdot p_A + \phi^A_{\mathrm{QCB}} + \phi^B_{\mathrm{QCB}} + \mathrm{gas}_B \cdot p_B + \delta(V) \tag{9.61}$$

Theorem 8 (Fee Bundling Savings)**.** *For uniform per-hop costs $\bar{c}$, the One-Hop-Swap achieves savings of:*

$$Savings = TCT(\pi_{multi}) - TCT(\pi_{OHS}) \geq (k-1) \cdot \bar{c}$$

Additionally, the reduced number of slippage events provides a compounding benefit:

$$V^{OHS}_{received} \geq V^{multi}_{received} \cdot \prod_{i=2}^{k-1} (1 - s_i)^{-1}$$

where s_i is the slippage fraction at hop i.

9.8.3 QCAI Fee Pre-Computation

Before a user commits to a transfer, QCAI pre-computes fee quotes for all available routes. The fee quote includes:

- Estimated source chain gas cost (based on current gas price oracle)
- Bridge protocol fee (fixed per endpoint + variable component proportional to volume)
- Estimated destination chain gas cost (QCAI prediction, see Equation 9.19)
- Expected slippage (based on current liquidity depth)
- Confidence interval on the total cost (95% bounds)

The bridge protocol fee for QCB is:

$$\phi^c_{\mathrm{QCB}} = \phi^c_{\mathrm{base}} + \phi_{\mathrm{variable}} \cdot V, \quad \phi_{\mathrm{variable}} \in [0.001, 0.005] \tag{9.62}$$

where ϕ^c_{base} is a per-chain fixed fee and ϕ_{variable} is the proportional fee rate, adjustable by governance.

9.8.4 Gas Abstraction for Bridge Fees

Through integration with the gas abstraction module, users can pay bridge fees in any accepted IBC-transferred token without holding native QOR. The gas abstraction conversion rate is:

$$\text{Fee}_{\text{token}} = \text{Fee}_{\text{QOR}} \cdot r_{\text{token}}/QOR \tag{9.63}$$

where $r_{\text{token/QOR}}$ is the governance-set exchange rate (e.g., $r_{\text{USDC/QOR}} = 1.0$, $r_{\text{ATOM/QOR}} = 10.0$).

```
# Get fee quote for a cross-chain transfer
qorechaind query bridge fee-quote \
  --source-chain ethereum \
  --dest-chain solana \
  --amount 1000000000 \
  --denom usdc \
  --output json

# Example response:
# {
#   "routes": [
#     {
#       "protocol": "qcb",
#       "estimated_fee_total": "2.34 USDC",
#       "breakdown": {
#         "source_gas": "0.42 USDC",
#         "bridge_fee": "1.50 USDC",
#         "dest_gas": "0.12 USDC",
#         "slippage_estimate": "0.30 USDC"
#       },
#       "confidence_95": ["2.10 USDC", "2.58 USDC"]
#     }
#   ]
# }

# Execute bridge transfer with gas abstraction (pay in USDC)
qorechaind tx bridge transfer \
  --from qor1abc...def \
  --dest-chain solana \
  --dest-address 7xKXtg2Cw...FkR \
  --amount 1000000000 \
  --denom usdc \
  --fee-denom ibc/USDC \
  --chain-id qorechain-diana
```

9.9 Cross-Chain Swap Optimisation

9.9.1 Liquidity Aggregation

QoreChain's hub position enables it to aggregate liquidity across all 25 connected chains. QCAI maintains a real-time liquidity map $\mathcal{L} : \mathcal{C} \times \mathcal{T} \to \mathbb{R}_{\geq 0}$, where $\mathcal{C}$ is the set of connected chains and $\mathcal{T}$ is the set of supported tokens. For a swap from token t_A on chain A to token t_B on chain B, QCAI solves:

$$\min_{\pi \in \Pi_{\text{swap}}} \Big[\text{Fee}(\pi) + \text{Slippage}(\pi, V) + \omega \cdot \text{Latency}(\pi) \Big] \tag{9.64}$$

subject to:

$$\mathcal{L}(c,t) \geq V_{\text{swap}} \quad \forall (c,t) \in \pi \tag{9.65}$$

$$S_{\text{security}}(\pi) \geq \theta_{\min} \tag{9.66}$$

where ω is the user's latency weight and $\theta_{\min}$ is the minimum acceptable security score for the swap path.

9.9.2 Slippage Model

For a constant-product automated market maker (AMM) with reserves (R_x, R_y), the slippage for a swap of amount Δx is:

$$\text{Slippage}(\Delta x) = 1 - \frac{R_y - \frac{R_x \cdot R_y}{R_x + \Delta x}}{\Delta x} \cdot \frac{R_y}{R_x} = \frac{\Delta x}{R_x + \Delta x} \tag{9.67}$$

For concentrated liquidity AMMs, the slippage depends on the active tick range and is computed as:

$$\text{Slippage}_{\text{CL}}(\Delta x) = \sum_i \frac{\Delta x_i}{L_i^2 / \sqrt{p_i \cdot \mathbb{1}[\Delta x_i > 0]}} \tag{9.68}$$

where L_i is the liquidity in tick range i and p_i is the price at tick i.

QCAI uses these models to predict slippage before execution and recommend optimal swap sizes. For large swaps, QCAI may recommend splitting the order across multiple paths to minimise aggregate slippage:

$$\min_{\{V_j \ \sum_{j=1}^{|\Pi|} \text{Slippage}_j(V_j) \quad \text{s.t.} \quad \sum_{j=1}^{|\Pi|} V_j = V_{\text{total}},\, V_j \geq 0} \tag{9.69}$$

9.9.3 Atomic Execution with Rollback

Cross-chain swaps execute atomically using a two-phase commitment protocol secured with PQC proofs:

Phase 1, Lock:

$$\text{Lock}(A, t_A, V) \xrightarrow{\text{PQC-signed proof}} \text{Commit}_{\mathcal{Q}}(\text{lock\_proof}) \tag{9.70}$$

Phase 2, Execute or Rollback:

$$\text{Execute}(B, t_B) = \begin{cases} \text{Release}(B, t_B, V') & \text{if Verify(lock\_proof)} = \top \ \wedge \ t < t_{\text{timeout}} \\ \text{Rollback}(A, t_A, V) & \text{otherwise} \end{cases} \tag{9.71}$$

The timeout t_{timeout} is set to accommodate the maximum expected latency across both bridge legs plus a safety margin:

$$t_{\text{timeout}} = \hat{\lambda}(A \to \mathcal{Q}) + \hat{\lambda}(\mathcal{Q} \to B) + 3\sigma_\lambda \tag{9.72}$$

where σ_λ is the standard deviation of the combined latency.

```
# Execute an atomic cross-chain swap
qorechaind tx bridge swap \
  --from qor1abc...def \
  --source-chain ethereum \
  --source-token ETH \
  --dest-chain solana \
  --dest-token SOL \
  --dest-address 7xKXtg2Cw...FkR \
  --amount 1000000000000000000 \
  --max-slippage 0.005 \
  --optimize-for fee \
  --chain-id qorechain-diana

# Query swap quote (preview without execution)
qorechaind query bridge swap-quote \
  --source-chain ethereum \
  --source-token ETH \
  --dest-chain solana \
  --dest-token SOL \
  --amount 1000000000000000000 \
  --output json

# Query swap execution status
qorechaind query bridge swap-status \
  --swap-id swap-00042 \
  --output json
```

9.10 Formal Security Analysis

9.10.1 Defence-in-Depth Model

The QoreChain bridge security architecture employs three independent defence layers, each providing a distinct security guarantee:

1. **AI Verification Layer ($\mathcal{D}_1$):** QCAI anomaly detection, fraud scoring, and cross-endpoint correlation (Section 9.4).
2. **PQC Attestation Layer ($\mathcal{D}_2$):** ML-DSA-87 signed attestations with quorum verification (Section 9.5).
3. **Circuit Breaker Layer ($\mathcal{D}_3$):** Volume-based circuit breakers and rate limiting (Section 9.7).

The probability of a successful bridge attack requires bypassing all three layers simultaneously:

$$P_{\text{attack}} = P(\text{bypass } \mathcal{D}_1) \cdot P(\text{bypass } \mathcal{D}_2) \cdot P(\text{bypass } \mathcal{D}_3) \tag{9.73}$$

Assuming layer independence:

$$P_{\text{attack}} \leq (1 - \text{AUC}_1) \cdot \text{negl}(\lambda) \cdot P(\text{volume} < \text{threshold}) \tag{9.74}$$

With $\text{AUC}_1 \geq 0.995$, $\lambda = 256$, and circuit breaker sensitivity $\kappa = 3$:

$$P_{\text{attack}} \leq 0.005 \cdot 2^{-128} \cdot 0.0013 \approx 2^{-138} \tag{9.75}$$

This bound is well below the 2^{-128} security target.

9.10.2 Byzantine Bridge Validator Tolerance

Theorem 9 (Bridge Safety)**.** *The QoreChain bridge is safe (no invalid transfers are finalised) as long as the number of Byzantine bridge validators f satisfies $f < \frac{n_c}{3}$ for each chain c, where n_c is the number of licensed validators for chain c.*

Proof sketch. A bridge transfer requires $Q_c = \lceil \frac{2}{3} n_c \rceil + 1$ valid attestations (Equation 9.40). For an invalid transfer to be accepted, an attacker must produce Q_c valid-looking attestations. With $f < \frac{n_c}{3}$ Byzantine validators, the attacker controls at most $f < \frac{n_c}{3}$ valid signing keys. Since $Q_c > \frac{2n_c}{3} > 2f$, the attacker cannot produce a quorum of attestations without cooperation from at least $Q_c - f > \frac{n_c}{3}$ honest validators, which by assumption will not attest to invalid events. □

The 1/3 Byzantine fault tolerance bound is optimal under the context of synchronous authenticated message passing (Dolev and Strong, 1983). Given the asynchronous nature of cross-chain bridging, QoreChain implements additional constraints: attestations must include a cryptographic commitment to the source chain's recent block header, time-stamped with QoreChain's block height. This prevents an attacker from retroactively forging attestations for historical events. Additionally, the bridge maintains an immutable audit log of all attestations; any two conflicting attestations from the same validator within the same epoch trigger automatic slashing, regardless of which attestation was accepted.

For a chain c with $n_c = 21$ licensed validators (a typical configuration), the fault tolerance is $f < 7$ Byzantine validators. The quorum requirement becomes $Q_c = 15$ attestations. With fewer than 7 compromised validators, an attacker controls at most 6 signing keys, leaving a gap of 9 attestations that must be forged or stolen from honest validators, which is cryptographically infeasible with ML-DSA-87 signatures. The BFT tolerance of 1/3 extends to the scenario where Byzantine validators attempt double-signing attacks; the time-stamped commitment prevents retroactive signature reuse, and the audit log ensures detection within a single block.

9.10.3 Game-Theoretic Analysis

Bridge validator incentives are designed to make honest behaviour the dominant strategy. Let r denote the per-epoch reward for honest bridge watching and s denote the slashing penalty for provably dishonest attestations. The payoff matrix for a validator choosing between honest (H) and dishonest (D) strategies is:

$$U(H) = r, \quad U(D) = \begin{cases} r + g & \text{with probability } P_{\text{undetected}} \\ -s & \text{with probability } 1 - P_{\text{undetected}} \end{cases} \tag{9.76}$$

where g is the potential gain from a successful attack. The expected utility of dishonesty is:

$$\mathbb{E}[U(D)] = P_{\text{undetected}} \cdot (r + g) + (1 - P_{\text{undetected}}) \cdot (-s) \tag{9.77}$$

Honest behaviour is the dominant strategy when $\mathbb{E}[U(H)] > \mathbb{E}[U(D)]$:

$$r > P_{\text{undetected}} \cdot (r + g) - (1 - P_{\text{undetected}}) \cdot s \tag{9.78}$$

Solving for the required slashing penalty:

$$s > \frac{P_{\text{undetected}} \cdot g - (1 - P_{\text{undetected}}) \cdot r}{1 - P_{\text{undetected}}} \tag{9.79}$$

With $P_{\text{undetected}} \leq 2^{-138}$ (from the defence-in-depth bound), even a modest slashing penalty renders dishonesty economically irrational for any realistic attack gain g.

9.10.4 Threat Model Summary

Table 9.4: Bridge Threat Model and Mitigations

Threat	**Defence Layer**	**Mitigation**
External chain compromise	$\mathcal{D}_2 + \mathcal{D}_3$	Confirmation depth + circuit breaker
Bridge validator collusion	$\mathcal{D}_1 + \mathcal{D}_2$	AI cross-correlation + quorum requirement
Relay manipulation	$\mathcal{D}_2$	PQC-signed attestations
Quantum adversary (future)	$\mathcal{D}_2$	ML-DSA-87 + ML-KEM-1024
Volume drain attack	$\mathcal{D}_1 + \mathcal{D}_3$	AI anomaly detection + circuit breaker
Coordinated multi-chain attack	$\mathcal{D}_1$	Cross-endpoint correlation analysis
Flash loan bridge exploit	$\mathcal{D}_1 + \mathcal{D}_3$	Anomaly detection + rate limiting + challenge period

```
# Query overall bridge security status
qorechaind query bridge security-status --output json

# Example response:
# {
#   "defence_layers": {
#     "ai_verification": {
#       "status": "active",
#       "auc_score": 0.997,
#       "transactions_scored_24h": 14523
#     },
#     "pqc_attestation": {
#       "status": "active",
#       "algorithm": "ML-DSA-87",
```

```
#       "total_licensed_validators": 42,
#       "chains_at_quorum": 17,
#       "chains_below_quorum": 0
#     },
#     "circuit_breaker": {
#       "endpoints_normal": 16,
#       "endpoints_triggered": 1,
#       "triggered_chains": ["polygon"]
#     }
#   },
#   "composite_security_score": 0.9994
# }

# Query bridge validator incentive parameters
qorechaind query bridge slashing-params --output json
```

Chapter 10

Combined Proof of Stake and AI-Driven Consensus

QoreChain's consensus mechanism extends the proven Cosmos SDK consensus foundation with three interconnected innovations: a triple-pool validator classification system that diversifies block production beyond pure stake dominance, an on-chain reinforcement learning agent that dynamically tunes consensus parameters in real time, and an optional quadratic-reputation-weighted governance extension, disabled at genesis and activatable by governance vote, designed to prevent plutocratic capture. This chapter formalises each component, its mathematical foundations, and its integration into the consensus pipeline.

10.1 QoreChain Consensus Engine Overview

10.1.1 Design Goals

The QoreChain Consensus Engine pursues four objectives simultaneously:

1. **Decentralisation beyond stake:** Pure stake-weighted selection concentrates block production among the wealthiest validators. QoreChain's triple-pool architecture ensures that reputation, delegation quality, and raw stake each contribute to proposer selection.

2. **Adaptive optimisation:** Network conditions change continuously. Rather than relying on static parameters, an on-chain RL agent observes chain state and adjusts block time, gas limits, and pool weights in real time.

3. **Quantum-safe validator operations:** All validator signatures, attestations, and governance votes are protected by ML-DSA-87 hybrid signatures, ensuring consensus integrity against future quantum adversaries.

4. **Fair governance:** At genesis, voting power is duration-weighted: staked QOR is scaled by lock-duration multipliers from 1.0x (no lock) to 2.0x (24-month lock), rewarding long-term commitment. An optional quadratic-reputation extension (QDRW, Section 10.8) can be activated by governance to further dampen raw capital accumulation.

Each goal addresses distinct failure modes observed in existing PoS networks. Concentration of block production in wealthy validators leads to plutocratic governance and vulnerability to large-scale miner-extractable value (MEV) extraction. Static parameters fail to adapt to emergent network conditions, such as sudden transaction volume spikes or validator churn. Classical signatures (ECDSA, EdDSA) remain vulnerable to cryptographic attacks by sufficiently advanced quantum computers, necessitating proactive migration to post-quantum cryptographic algorithms. Governance systems weighted purely by token holdings create perverse incentives where large holders prioritise short-term gains over long-term network health. QoreChain's design philosophy integrates these four objectives into a coherent consensus protocol that balances security, fairness, and adaptability.

10.1.2 Consensus Pipeline

The per-block consensus pipeline operates as follows:

1. **Observation:** The RL agent records a 25-dimension state vector capturing current chain conditions (transaction volume, gas utilisation, block time variance, validator participation, bridge activity, PQC verification load).
2. **Parameter Tuning:** The RL agent computes and applies parameter adjustments (block time target, gas limit, gas price floor, pool weights) subject to safety bounds and circuit breaker checks.
3. **Pool Classification:** Every 1,000 blocks, validators are reclassified into RPoS, DPoS, or PoS pools based on their delegation ratio and reputation score.
4. **Proposer Selection:** The next block proposer is selected via pool-weighted sortition, where each validator's selection probability depends on their pool's weight and their individual reputation.
5. **Block Production:** The selected proposer constructs the block, applying the 5-lane transaction prioritisation (PQC, MEV, AI, Default, Free).
6. **Voting and Finalisation:** Validators vote on the proposed block using hybrid PQC signatures. The block is finalised when $> 2/3$ of voting power agrees.
7. **Reward Distribution:** Bonding curve rewards are computed for the proposer and participating validators, incorporating stake, loyalty, and reputation.

The formal safety and liveness conditions are:

$$\textbf{Safety:} \quad \forall h,\ \nexists\, B_1, B_2 : B_1 \neq B_2 \ \wedge\ \text{Finalised}(B_1, h) \ \wedge\ \text{Finalised}(B_2, h) \tag{10.1}$$

$$\textbf{Liveness:} \quad \text{If } |\{v : \text{honest}(v)\}| > \frac{2}{3} \cdot |V|,\ \text{then } \forall t,\ \exists h > h_t : \text{Finalised}(B, h) \tag{10.2}$$

```
# Query current consensus parameters
qorechaind query consensus params --output json

# Query the consensus pipeline status
qorechaind query qca status --output json

# Example response:
# {
#   "current_block": 142857,
#   "rl_agent_mode": "conservative",
#   "current_epoch": 1428,
#   "pool_weights": {
#     "rpos": "0.42",
#     "dpos": "0.34",
#     "pos": "0.24"
#   },
#   "last_classification_height": 142000,
#   "next_classification_height": 143000,
#   "active_validators": 87,
#   "target_block_time": "5.2s"
# }
```

10.2 Combined Proof of Stake: Triple-Pool Architecture

10.2.1 Pool Classification

QoreChain's Combined Proof of Stake (CPoS) partitions the active validator set into three pools, each emphasising a different dimension of validator quality. Let $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$ denote the active validator set. For each validator v_i, define:

- $d_i = \frac{\text{delegated\_stake}(v_i)}{\text{total\_stake}(v_i)}$: the delegation ratio (fraction of stake from external delegators)
- $r_i \in [0, 1]$: the reputation score (see Section 10.3)

Every $I_{\text{class}} = 1{,}000$ blocks, the pool classifier assigns each validator to exactly one pool:

$$\text{Pool}(v_i) = \begin{cases} \text{DPoS} & \text{if } d_i \geq \mathcal{P}_{75}(\{d_j\}_{j=1}^{n}) \\ \text{RPoS} & \text{if } r_i \geq \mathcal{P}_{75}(\{r_j\}_{j=1}^{n}) \ \wedge \ \text{Pool}(v_i) \neq \text{DPoS} \\ \text{PoS} & \text{otherwise} \end{cases} \tag{10.3}$$

where $\mathcal{P}_{75}(\cdot)$ denotes the 75th percentile of the given set. The classification is deterministic and produces the same result on all validators given the same state.

10.2.2 Pool Weights and Proposer Selection

Each pool carries a base weight that determines its share of block proposals:

Table 10.1: CPoS Pool Configuration

Pool	Weight (w_p)	Selection Emphasis	Typical Share
RPoS (Reputation)	0.40	Long-term honest behaviour	40% of proposals
DPoS (Delegation)	0.35	Community trust via delegation	35% of proposals
PoS (Standard)	0.25	Pure stake weight	25% of proposals

Note that the RL agent may adjust these weights dynamically within bounded ranges (Section 10.6).

The proposer selection probability for validator v_i in pool p is:

$$\Pr[\text{proposer} = v_i] = \frac{w_p \cdot r_i \cdot s_i}{\sum_{j=1}^{n} w_{p(j)} \cdot r_j \cdot s_j} \tag{10.4}$$

where w_p is the weight of validator v_i's pool, r_i is the reputation score, and s_i is the normalised stake. This formula ensures that all three dimensions contribute to selection probability.

10.2.3 Decentralisation Analysis

To quantify the decentralisation improvement of CPoS over pure stake-weighted selection, we use the Gini coefficient. For a set of proposer probabilities $\{p_1, p_2, \ldots, p_n\}$ sorted in ascending order:

$$G = \frac{2\sum_{i=1}^{n} i \cdot p_i}{n\sum_{i=1}^{n} p_i} - \frac{n+1}{n} \tag{10.5}$$

$G = 0$ represents perfect equality (every validator has equal selection probability) and $G = 1$ represents maximum inequality (one validator produces all blocks).

Theorem 10 (CPoS Decentralisation Bound)**.** *Let G_{stake} be the Gini coefficient under pure stake-weighted selection and G_{CPoS} under CPoS with pool weights (w_R, w_D, w_S). If the reputation and delegation distributions are not perfectly correlated with stake:*

$$G_{CPoS} \leq G_{stake} - \eta$$

where $\eta > 0$ depends on the cross-correlation between stake, reputation, and delegation ratio. In the extreme case where reputation and delegation are uniformly distributed:

$$\eta = G_{stake} \cdot \left(1 - \frac{w_S}{w_R + w_D + w_S}\right) \cdot (1 - \rho_{sr})$$

where ρ_{sr} is the Pearson correlation between stake and reputation.

The Nakamoto coefficient (minimum number of entities controlling $> 1/3$ of selection probability) also improves under CPoS:

$$N_{\text{CPoS}} = \min\left\{k : \sum_{i=n-k+1}^{n} \Pr[\text{proposer} = v_i] > \frac{1}{3}\right\} \tag{10.6}$$

With three independent weighting dimensions, the top-stake validators no longer dominate unless they also hold top reputation and top delegation ratio.

```
# Query pool classification for all validators
qorechaind query qca pool-classifications --output json

# Example response:
# {
#   "classifications": [
#     {
#       "validator": "qorvaloper1abc...def",
#       "pool": "RPoS",
#       "stake": "5000000000000",
#       "delegation_ratio": "0.45",
#       "reputation_score": "0.92",
#       "classified_at_height": 142000
#     }
#   ],
#   "pool_summary": {
#     "rpos_count": 22,
#     "dpos_count": 28,
#     "pos_count": 37
#   }
# }

# Query a specific validator's pool assignment
qorechaind query qca pool-classification \
  --validator qorvaloper1abc...def

# Query current Gini coefficient and Nakamoto coefficient
qorechaind query qca decentralisation-metrics --output json
```

10.3 Validator Reputation System

10.3.1 Multi-Dimensional Scoring

The reputation score R_i for validator v_i is a weighted combination of four dimensions:

$$R_i = \alpha \cdot S_i + \beta \cdot P_i + \gamma \cdot C_i + \delta \cdot T_i \tag{10.7}$$

where:

- $S_i \in [0, 1]$: **Stake weight** (normalised). $S_i = s_i / \max_j(s_j)$, reflecting the validator's economic commitment.

- $P_i \in [0,1]$: **Performance score**. Computed from uptime fraction u_i and block production rate b_i:

$$P_i = \frac{u_i + b_i}{2}, \quad u_i = \frac{\text{signed} blocks}{\text{total blocks in} window}, \quad b_i = \frac{\text{produced} blocks}{\text{expected} blocks} \tag{10.8}$$

- $C_i \in [0,1]$: **Community trust score**. Derived from delegation diversity (number of unique delegators) and governance participation rate.

- $T_i \in [0,1]$: **Temporal factor**. Rewards longevity: $T_i = 1 - e^{-\mu \cdot \tau_i}$ where τ_i is the number of blocks since the validator joined the active set and μ is a decay constant.

The weights $(\alpha, \beta, \gamma, \delta)$ are governance-configurable and sum to 1. Default values emphasise performance and community trust over raw stake.

10.3.2 Temporal Decay

Reputation scores decay over time to ensure that past good behaviour does not indefinitely shield a validator that has become unreliable. The reputation at block h is:

$$R_i(h) = R_i^{\text{base}}(h) \cdot e^{-\kappa(h - h_{\text{last}})} \tag{10.9}$$

where $R_i^{\text{base}}(h)$ is the freshly computed score at block h, h_{last} is the height of the most recent reputation update, and κ is the decay rate. Reputation is recomputed periodically (every epoch), so the decay serves as a penalty for validators that miss evaluation windows.

The decay mechanism is triggered when a validator has not participated in consensus for a sustained period. Let $w_{\text{inactivity}}$ denote the inactivity threshold in blocks; decay begins when the validator's participation gap exceeds this window. The reputation half-life is:

$$t_{1/2} = \frac{\ln 2}{\kappa} \tag{10.10}$$

At default settings ($\kappa = 0.001$ per block), the half-life is approximately 693 blocks, corresponding to roughly 68 minutes at 6-second block time. This ensures fast recovery for transient outages while accumulating penalties for chronic underperformance. Once reputation falls below the minimum threshold (default 0.2), the validator is automatically ejected from the active set and must undergo full re-staking and evaluation to rejoin.

Between evaluation epochs, reputation decays exponentially according to Equation 10.9. At epoch boundaries (every E blocks, typically 100 blocks), the system recomputes $R_i^{\text{base}}(h)$ from fresh consensus metrics: uptime, proposal quality, and finality participation. This dual-mode mechanism prevents unbounded reputation losses from brief disconnections while penalising sustained misbehaviour. Validators with complete participation maintain steady reputation, whereas those with frequent absences experience cumulative reputation attrition across epochs.

10.3.3 Reputation Multiplier

The raw reputation score $r_i \in [0, 1]$ is transformed into a multiplier $M(r_i)$ used throughout the consensus system (proposer selection, bonding curve, governance). The transformation uses a sigmoid function:

$$M(r_i) = 0.5 + 1.5 \cdot \sigma\Big(6 \cdot (r_i - 0.5)\Big) \tag{10.11}$$

where $\sigma(x) = \frac{e^x}{1+e^x}$ is the logistic sigmoid function. This maps reputation to the range $[0.5, 2.0]$:

- A validator with $r_i = 0$ receives a multiplier of approximately 0.51 (halved influence).
- A validator with $r_i = 0.5$ receives a multiplier of 1.25 (neutral).
- A validator with $r_i = 1.0$ receives a multiplier of approximately 1.99 (doubled influence).

The sigmoid shape ensures smooth transitions and prevents discontinuities at the boundaries. The steepness parameter (6) is chosen so that most of the transition occurs in the $[0.3, 0.7]$ range, rewarding clear differentiation while compressing the extremes.

$$\frac{dM}{dr_i} = 9 \cdot \sigma\Big(6(r_i - 0.5)\Big) \cdot \Big(1 - \sigma(6(r_i - 0.5))\Big) \tag{10.12}$$

The maximum sensitivity is at $r_i = 0.5$, where $\frac{dM}{dr_i} = 9 \cdot 0.25 = 2.25$, meaning a 1% improvement in reputation yields a 2.25% improvement in multiplier at the midpoint.

```
# Query validator reputation score
qorechaind query reputation score \
  --validator qorvaloper1abc...def \
  --output json

# Example response:
# {
#   "validator": "qorvaloper1abc...def",
#   "reputation_score": "0.87",
#   "components": {
#     "stake_weight": "0.72",
#     "performance": "0.95",
#     "community_trust": "0.84",
#     "temporal_factor": "0.91"
#   },
#   "multiplier": "1.82",
#   "pool_assignment": "RPoS",
#   "last_updated_height": 142850
# }

# Query reputation parameters
qorechaind query reputation params
```

10.4 Bonding Curve Reward Model

10.4.1 Reward Formula

Validator rewards in QoreChain go beyond simple stake-proportional distribution. The bonding curve reward function incorporates four factors: stake magnitude, delegation loyalty, reputation quality, and network phase:

$$R(v,t) = \beta \cdot S_v \cdot \Big(1 + \alpha \cdot \ln(1 + L_v)\Big) \cdot Q(r_v) \cdot P(t) \tag{10.13}$$

where:

- β: Base reward coefficient (default: 1.0)
- S_v: Stake of validator v (in QOR)
- α: Loyalty sensitivity parameter (default: 0.1)
- L_v: Loyalty metric, defined as the weighted average delegation duration across all delegators of v
- $Q(r_v)$: Reputation quality factor
- $P(t)$: Phase multiplier

10.4.2 Loyalty Component

The logarithmic loyalty term $\ln(1 + L_v)$ rewards long-term delegation while exhibiting diminishing returns. A validator whose delegators have been staking for an average of L_v epochs receives a bonus that grows logarithmically:

$$\text{LoyaltyBonus}(L_v) = \alpha \cdot \ln(1 + L_v) \tag{10.14}$$

For $\alpha = 0.1$:

- $L_v = 0$ epochs: bonus $= 0.0$ (no loyalty history)
- $L_v = 10$ epochs: bonus $= 0.1 \cdot \ln(11) \approx 0.024$ (+2.4%)
- $L_v = 100$ epochs: bonus $= 0.1 \cdot \ln(101) \approx 0.046$ (+4.6%)
- $L_v = 1000$ epochs: bonus $= 0.1 \cdot \ln(1001) \approx 0.069$ (+6.9%)

The logarithm is computed using a deterministic Taylor series approximation to ensure identical results across all validators:

$$\ln(1 + x) = \sum_{k=1}^{15} \frac{(-1)^{k+1}}{k} \cdot x^k \quad \text{(with argument reduction for } x > 1.5) \tag{10.15}$$

For $x > 1.5$, argument reduction divides x by 2 repeatedly until $x \leq 1.5$, computes the Taylor approximation, and adds $n \cdot \ln(2)$ where n is the number of halvings.

10.4.3 Reputation Quality Factor

The quality factor $Q(r_v)$ maps the reputation score to a multiplier clamped to the range $[0.75, 1.25]$:

$$Q(r_v) = \text{clamp}\Big(0.5 + r_v,\ 0.75,\ 1.25\Big) \tag{10.16}$$

This ensures that validators with poor reputation ($r_v < 0.25$) are penalised by up to 25%, while validators with excellent reputation ($r_v > 0.75$) receive up to a 25% bonus. The clamping prevents extreme rewards or penalties that could destabilise the validator set.

The quality factor applies directly to block proposal selection, reward accrual, and delegation incentive multipliers. A validator with $Q = 0.75$ is selected as proposer with 75% frequency relative to baseline, reducing earning potential proportionally. Validators with $Q \geq 1.0$ receive enhanced proposal selection and higher rewards per block, creating positive feedback for sustained performance.

Calculation of r_v employs a rolling 100-block window of four consensus metrics: uptime (blocks signed out of total), finality rate (times participated in successful finality out of opportunities), proposal quality (measured as inclusion rate for valid transactions), and governance engagement (delegation diversity and on-chain votes). The base reputation is the unweighted average of these four metrics, each normalised to [0, 1].

When validators exhibit sudden performance degradation after periods of high reputation, the quality factor adjusts immediately to prevent reward windfall. The clamp bounds are enforced invariantly across all reward distributions to prevent pathological scenarios where a single validator dominates earning. The floor at 0.75 ensures underperforming validators can still earn and gradually recover; the ceiling at 1.25 prevents extreme concentration of validator earnings and maintains long-term network decentralisation.

10.4.4 Phase Multiplier

The phase multiplier $P(t)$ adjusts the overall reward level based on the network's maturity stage:

Table 10.2: Phase Multiplier Schedule

Phase	$P(t)$	**Purpose**
Genesis (Year 1)	1.5	Aggressive incentives for early validators
Growth (Year 2)	1.0	Standard rewards as network matures
Mature (Year 3+)	0.8	Reduced emissions, offset by fee revenue

10.4.5 Reward Sensitivity Analysis

The partial derivatives of $R(v, t)$ with respect to each parameter reveal the marginal impact of each factor:

$$\frac{\partial R}{\partial S_v} = \beta \cdot (1 + \alpha \ln(1 + L_v)) \cdot Q(r_v) \cdot P(t) \tag{10.17}$$

$$\frac{\partial R}{\partial L_v} = \beta \cdot S_v \cdot \frac{\alpha}{1 + L_v} \cdot Q(r_v) \cdot P(t) \tag{10.18}$$

$$\frac{\partial R}{\partial r_v} = \beta \cdot S_v \cdot (1 + \alpha \ln(1 + L_v)) \cdot Q'(r_v) \cdot P(t) \tag{10.19}$$

Key insight: the loyalty sensitivity $\frac{\partial R}{\partial L_v}$ decays as $O(1/L_v)$ for large L_v, ensuring diminishing returns and preventing infinite reward accumulation from loyalty alone. Meanwhile, the stake sensitivity $\frac{\partial R}{\partial S_v}$ is approximately linear, maintaining the economic incentive to stake.

```
# Query bonding curve parameters
qorechaind query qca bonding-curve-params --output json

# Estimate reward for a specific validator
qorechaind query qca estimate-reward \
  --validator qorvaloper1abc...def \
  --output json

# Example response:
# {
#   "validator": "qorvaloper1abc...def",
#   "estimated_reward_per_epoch": "4523.87 QOR",
#   "components": {
#     "stake": "5000000",
#     "loyalty_bonus": "0.046",
#     "quality_factor": "1.18",
#     "phase_multiplier": "1.0"
#   },
#   "current_phase": "growth"
# }
```

10.5 Progressive Slashing

10.5.1 Escalating Penalty Model

Traditional blockchain slashing applies fixed penalties regardless of a validator's history. QoreChain implements progressive slashing: penalties escalate with repeated infractions but decay over time, giving validators an incentive to improve behaviour rather than exit the network.

The penalty for an infraction at block h is:

$$\text{Penalty}(v, h) = \min\left(\text{base\_rate} \cdot \text{escalation}^{E_{\text{eff}}(v,h)} \cdot \text{severity}(f),\ \ P_{\max}\right) \tag{10.20}$$

where:

- base_rate $= 0.01$ (1% for a first-time minor infraction)
- escalation $= 1.5$ (each effective prior infraction increases the penalty by 50%)

- $E_{\text{eff}}(v, h)$ is the effective infraction count with temporal decay (see below)
- $\text{severity}(f) \in [0.1, 10.0]$ depends on the infraction type f
- $P_{\max} = 0.33$ (maximum 33% penalty per infraction)

10.5.2 Temporal Decay

Prior infractions contribute to the effective count with exponentially decaying weight:

$$E_{\text{eff}}(v, h) = \sum_{k=1}^{K} \exp\left(-\frac{\ln 2 \cdot (h - h_k)}{H_{1/2}}\right) \tag{10.21}$$

where h_k is the height of the k-th prior infraction and $H_{1/2} = 100{,}000$ blocks is the half-life. The half-life property ensures:

$$\text{weight}(h_k, h) = \begin{cases} 1.0 & \text{if } h - h_k = 0 \\ 0.5 & \text{if } h - h_k = H_{1/2} \\ 0.25 & \text{if } h - h_k = 2H_{1/2} \\ \to 0 & \text{as } h - h_k \to \infty \end{cases} \tag{10.22}$$

This means a validator who committed an infraction 100,000 blocks ago has that infraction count as only 0.5 towards their effective count, and one from 200,000 blocks ago counts as only 0.25.

10.5.3 Infraction Types and Severity

Table 10.3: Infraction Types and Base Penalties

Infraction	Severity	First-Offence Penalty	Description
Downtime	1.0	0.01%	Missing $> k$ consecutive blocks
Late voting	0.5	0.005%	Consistently late consensus votes
Double-sign	5.0	5.0%	Signing conflicting blocks at same height
Invalid attestation	3.0	3.0%	Producing invalid bridge attestation

10.5.4 Comparison with Flat-Penalty Models

Under a flat-penalty model with fixed penalty p_{flat}, the expected penalty over T blocks for a validator with infraction rate λ is:

$$\mathbb{E}[\text{TotalPenalty}_{\text{flat}}] = \lambda T \cdot p_{\text{flat}} \tag{10.23}$$

Under progressive slashing, the expected penalty is:

$$\mathbb{E}[\text{TotalPenalty}_{\text{prog}}] = \sum_{k=1}^{\lambda T} \min\left(p_{\text{base}} \cdot \text{esc}^{E_{\text{eff}}^{(k)}}, P_{\max}\right) \tag{10.24}$$

For validators with low infraction rates ($\lambda \ll 1/H_{1/2}$), progressive slashing is more lenient than flat penalties (infractions decay before the next one occurs). For validators with high infraction rates ($\lambda \gg 1/H_{1/2}$), progressive slashing escalates rapidly, reaching

$P_{\max}$ and becoming significantly harsher than flat penalties. This creates a strong incentive gradient: occasional mistakes are tolerated, but persistent misbehaviour is severely punished.

$$\frac{\mathbb{E}[\text{TotalPenalty}_{\text{prog}}]}{\mathbb{E}[\text{TotalPenalty}_{\text{flat}}]} \approx \begin{cases} < 1 & \text{if } \lambda \cdot H_{1/2} < 1 \text{ (rare infractions)} \\ > 1 & \text{if } \lambda \cdot H_{1/2} > 2 \text{ (frequent infractions)} \end{cases} \tag{10.25}$$

```
# Query slashing parameters
qorechaind query qca slashing-params --output json

# Query slashing history for a validator
qorechaind query qca slashing-records \
  --validator qorvaloper1abc...def \
  --output json

# Example response:
# {
#   "validator": "qorvaloper1abc...def",
#   "records": [
#     {
#       "infraction_height": 130000,
#       "infraction_type": "downtime",
#       "severity_factor": "1.0",
#       "penalty_applied": "0.01",
#       "effective_count_at_time": "0.0"
#     }
#   ],
#   "current_effective_count": "0.37",
#   "next_penalty_estimate": {
#     "downtime": "0.0145",
#     "double_sign": "7.25"
#   }
# }
```

10.6 On-Chain Reinforcement Learning Agent

10.6.1 Architecture

QoreChain's reinforcement learning consensus agent, **PRISM** (Policy-driven Reinforcement-learning for Intelligent State Machines), is structured as two cooperating roles that share a single multi-layer perceptron (MLP) policy: (1) an **inline inference path** embedded directly in the block lifecycle, which applies the current policy on every block to dynamically tune consensus parameters; and (2) the **PRISM Trainer**, an off-path process that observes network state on a slower cadence, computes PPO gradient updates, and periodically publishes refreshed weights back to the inline path via an on-chain `MsgUpdatePolicy` transaction (further specified in Section 15.3.5). This section describes the inline inference path. The path runs a Go-native fixed-point

MLP that executes a forward pass on every block using the currently published policy weights. The inference call itself requires no external oracle, no off-chain RPC, and no off-chain computation; the trained weights are the only input from the PRISM Trainer, and they are delivered through the on-chain `MsgUpdatePolicy` transaction path rather than any out-of-band channel.

The MLP architecture is:

$$\text{MLP} : \mathbb{R}^{25} \xrightarrow{\mathbf{W}_1,\mathbf{b}_1} \mathbb{R}^{256} \xrightarrow{\text{ReLU}\mathbb{R}^{256} \xrightarrow{\mathbf{W}_2,\mathbf{b}_2} \mathbb{R}^{256} \xrightarrow{\text{ReLU}\mathbb{R}^{256} \xrightarrow{\mathbf{W}_3,\mathbf{b}_3} \mathbb{R}^{5} \xrightarrow{\tanh} [-1,1]^5} \quad (10.26)$$

Total parameter count:

$$|\theta| = (25 \times 256 + 256) + (256 \times 256 + 256) + (256 \times 5 + 5) = 73{,}733 \quad (10.27)$$

The 25-dimensional input state vector captures: (1) transaction pool size (num pending transactions), (2) average transaction fee (in uqor), (3) block propagation latency (95th percentile in ms), (4) validator participation rate (fraction of validators in-sync), (5) bridge activity level (cross-chain transactions per block), (6) PQC verification overhead (time spent verifying signatures), (7-11) per-pool validator counts (RPoS, DPoS, PoS, Light, Unclassified), (12-16) per-pool average reputation scores, (17-21) per-pool stake distributions, (22) current block time (seconds), (23) current gas limit, (24) recent MEV extraction magnitude, and (25) network partition detector (boolean flag from monitoring system). The 5-dimensional output represents adjustments to: (1) block time target offset, (2) gas limit multiplier, (3) base gas price adjustment, (4) RPoS pool weight, and (5) DPoS pool weight. All outputs are clamped to safe ranges; block time cannot deviate more than 20% from the protocol baseline, gas limits cannot exceed twice the previous block's limit, and pool weights must remain positive and sum to 1.

10.6.2 Deterministic Fixed-Point Arithmetic

Consensus-critical computation must produce identical results on every validator. The MLP uses integer arithmetic at scale 10^8 (8 decimal places of precision) with no floating-point operations:

$$\hat{x} = \lfloor x \times 10^8 \rceil, \quad \hat{x} \cdot \hat{y} = \left\lfloor \frac{\hat{x} \times \hat{y}}{10^8} \right\rceil \quad (10.28)$$

All activation functions are approximated using Taylor series:

$$\tanh(x) \approx x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots \quad \text{(7-term series, clamped to } [-1, 1]) \quad (10.29)$$

$$e^x \approx \sum_{k=0}^{12} \frac{x^k}{k!} \quad \text{(12-term Taylor series)} \quad (10.30)$$

$$\sigma(x) = \frac{e^x}{1 + e^x} \quad \text{(computed via ExpApprox with negative-}x\text{ symmetry)} \quad (10.31)$$

The deterministic math library provides five functions: `IntegerSqrt` (Newton's method, 100 iterations), `TaylorLn1PlusX` (argument reduction + 15-term series), `SigmoidApprox`, `ReputationMultiplier`, and `ExpApprox`. All use the Cosmos SDK `LegacyDec` type, ensuring byte-identical results across all validator nodes.

10.6.3 Observation Vector

The RL agent observes a 25-dimension state vector every I_{obs} blocks (default: 10):

$$\mathbf{o}(t) = (o_1, o_2, \ldots, o_{25}) \in \mathbb{Z}^{25} \tag{10.32}$$

Table 10.4: RL Agent Observation Dimensions

Dim	Name	Description
1	tx_count	Transactions in observation interval
2	avg_tx_size	Mean transaction size (bytes)
3	gas_utilisation	Fraction of gas limit consumed
4	block_time	Actual block time vs target
5	pending_tx_count	Mempool depth
6	validator_count	Total registered validators
7	active_validators	Validators that signed recent blocks
8	missed_blocks	Blocks missed in interval
9	avg_reputation	Mean reputation across active set
10	block_reward	Total rewards distributed
11	total_fees	Total fees collected
12	mempool_size	Mempool byte size
13–14	gas prices	Average and maximum gas price
15	block_size	Average block size (bytes)
16	cross_vm_msgs	Cross-VM messages processed
17	bridge_txs	Bridge transactions processed
18	pqc_verifications	PQC signature verifications
19	ai_verdicts	AI verdict evaluations
20	reputation_updates	Reputation score changes
21	unique_senders	Distinct transaction signers
22	avg_tx_latency	Mean time from submission to inclusion
23	chain_load	Composite load indicator
24	consensus_rounds	Rounds to reach consensus
25	proposer_diversity	Unique proposers in interval

10.6.4 Action Space

The MLP output is a 5-dimension action vector, each component in $[-1, 1]$, mapped to parameter adjustments:

Table 10.5: RL Agent Action Space

Dim	Parameter	Max Adjustment	Safe Bounds
1	Block time target	± configurable	≥ 1 s
2	Gas limit	± configurable	$\geq$ 1,000,000
3	Gas price floor	± configurable	≥ 1
4	RPoS pool weight	± configurable	$[0.1, 0.8]$
5	DPoS pool weight	± configurable	$[0.1, 0.8]$

The PoS pool weight is derived as $w_{\text{PoS}} = 1 - w_{\text{RPoS}} - w_{\text{DPoS}}$, constrained to $[0.1, 0.8]$.

10.6.5 Multi-Objective Reward Function

After each block is finalised, the agent receives a reward signal with five components:

$$r(t) = \lambda_1 \cdot r_{\text{throughput}} + \lambda_2 \cdot r_{\text{finality}} + \lambda_3 \cdot r_{\text{decentralisation}} + \lambda_4 \cdot r_{\text{MEV}} + \lambda_5 \cdot r_{\text{failure}} \quad (10.33)$$

where:

- $r_{\text{throughput}} = \frac{\text{tx\_count}(I_{\text{obs}})}{\text{tx\_capacity}}$: normalised transaction throughput
- $r_{\text{finality}} = -|\text{block\_time} - \text{target\_block\_time}|$: penalty for block time deviation
- $r_{\text{decentralisation}} = \frac{\text{unique\_proposers}(I_{\text{obs}})}{\text{interval\_length}}$: proposer diversity in interval
- $r_{\text{MEV}} = -\text{Var}(\text{gas\_prices})$: penalty for gas price variance (proxy for MEV extraction)
- $r_{\text{failure}} = -\frac{\text{failed\_txs}}{\text{total\_txs}}$: penalty for transaction failure rate

The PPO objective that the agent optimises over training epochs is:

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t,\ \text{clip} \left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}, 1-\epsilon, 1+\epsilon \right) \hat{A}_t \right) \right] \quad (10.34)$$

where $\hat{A}_t$ is the advantage estimate and ϵ is the clipping parameter (default: 0.2).

10.6.6 Operational Modes

The RL agent operates in one of four governance-controlled modes:

Table 10.6: RL Agent Operational Modes

Mode	Behaviour
Shadow	Observe and record state; no parameter changes. Used for data collection during initial deployment.
Conservative	Apply actions with 50% dampening: $a_{\text{applied}} = 0.5 \cdot a_{\text{raw}}$. Limits the impact of potentially suboptimal policy.
Autonomous	Full RL-driven adjustments. The agent's actions are applied directly (within safe bounds).
Paused	No-op. The agent is inactive; all parameters remain at their current values.

Mode transitions require a governance proposal, ensuring that the validator community controls the agent's autonomy level. The recommended deployment progression is: Shadow (50 epochs) → Conservative → Autonomous.

```
# Query RL agent status
qorechaind query rlconsensus agent-status --output json

# Example response:
# {
#   "mode": "conservative",
#   "epoch": 1428,
#   "circuit_breaker_active": false,
#   "observation_interval": 10,
#   "shadow_mode_epochs": 50,
#   "mlp_parameters": 73733,
#   "last_action": {
#     "block_time_adj": "-0.02",
#     "gas_limit_adj": "0.05",
#     "gas_price_floor_adj": "0.00",
#     "rpos_weight_adj": "0.01",
#     "dpos_weight_adj": "-0.01"
#   }
# }

# Query latest observation vector
qorechaind query rlconsensus observation --output json

# Query latest reward breakdown
qorechaind query rlconsensus reward --output json

# Change agent mode (governance tx)
qorechaind tx rlconsensus set-mode autonomous \
  --from qor1governance...addr \
  --chain-id qorechain-diana
```

10.7 RL Circuit Breaker

10.7.1 Safety Mechanism

The RL agent is a powerful but potentially dangerous component: a malfunctioning policy could degrade chain performance by setting adversarial parameters. The circuit breaker provides an automatic safety net that activates when chain health deteriorates.

The circuit breaker monitors the on-time block ratio over a sliding window of $W = 50$ blocks:

$$\text{OnTimeRatio}(t) = \frac{|\{h \in [t - W + 1, t] : \text{block\_time}(h) \leq \text{target}\}|}{W} \tag{10.35}$$

The circuit breaker triggers when:

$$\text{OnTimeRatio}(t) < \theta_{\text{CB}}, \quad \theta_{\text{CB}} = 0.5 \tag{10.36}$$

meaning fewer than half of the blocks in the window met the target block time.

10.7.2 Circuit Breaker Response

When triggered, the circuit breaker executes a three-step recovery that prioritizes safety over continued operation. The recovery mechanism is designed to interrupt destabilizing feedback loops while maintaining the ability for governance to assess root causes and decide on corrective actions. Recovery is automatic to minimize human intervention latency in crisis scenarios, reducing the time between anomaly detection and corrective response from minutes to seconds. The three-step process creates cascading safety levels: parameter reversion eliminates the cause of observed anomaly, agent pause prevents recursive reconfiguration during recovery, and event emission ensures all stakeholders are informed and can prepare coordinated responses.

1. **Parameter Revert:** All RL-adjusted parameters (block time, gas limit, gas price floor, pool weights) are immediately reverted to their default values.
2. **Agent Pause:** The RL agent is set to Paused mode for W blocks, preventing any further parameter adjustments during recovery.
3. **Event Emission:** A `circuit_breaker_triggered` event is emitted, alerting monitoring systems and governance participants.

After W blocks in Paused mode, the agent auto-resumes in the mode it was in before the trigger (typically Conservative or Autonomous).

10.7.3 False Trigger Analysis

Under normal operating conditions where block times follow a distribution with mean μ and standard deviation σ around the target, the probability of the on-time ratio falling below θ_{CB} is:

$$\Pr[\text{false trigger}] = \Pr\left[\text{Bin}(W, p_{\text{ontime}}) < \theta_{\text{CB}} \cdot W\right] \tag{10.37}$$

where p_{ontime} is the probability that a single block meets the target time. For $p_{\text{ontime}} = 0.9$ (90% of blocks on time), $W = 50$, and $\theta_{\text{CB}} = 0.5$:

$$\Pr[\text{false trigger}] = \sum_{k=0}^{24} \binom{50}{k} (0.9)^k (0.1)^{50-k} < 10^{-10} \tag{10.38}$$

The false trigger rate is negligible under normal conditions, ensuring that the circuit breaker activates only during genuine degradation.

10.7.4 Recovery Time

The expected time from trigger to full recovery is:

$$T_{\text{recovery}} = W \cdot \bar{t}_{\text{block}} + T_{\text{stabilise}} \tag{10.39}$$

where $\bar{t}_{\text{block}}$ is the average block time under default parameters and $T_{\text{stabilise}}$ is the time for the agent to re-adapt after resuming. With $W = 50$ and $\bar{t}_{\text{block}} \approx 5$ s, the minimum recovery time is approximately 250 seconds (about 4 minutes).

```
# Query circuit breaker status
qorechaind query rlconsensus circuit-breaker --output json

# Example response (when triggered):
# {
#   "active": true,
#   "triggered_at_block": 142800,
#   "on_time_ratio_at_trigger": "0.42",
#   "blocks_remaining": 23,
#   "reverted_params": {
#     "block_time": "5.0s",
#     "gas_limit": "50000000",
#     "rpos_weight": "0.40",
#     "dpos_weight": "0.35"
#   },
#   "previous_mode": "autonomous"
# }

# Resume agent after manual investigation (governance tx)
qorechaind tx rlconsensus resume \
  --from qor1governance...addr \
  --chain-id qorechain-diana
```

10.8 Quadratic-Delegation Reputation-Weighted Governance

10.8.1 Voting Power Formula

Activation status. At genesis, governance voting power is duration-weighted, consistent with the QoreChain Tokenomics Paper: staked QOR is scaled by a lock-duration

multiplier of 1.0x (no lock), 1.2x (3-month lock), 1.4x (6 months), 1.7x (12 months), or 2.0x (24 months), applied by the xQORE module. The quadratic-reputation-weighted (QDRW) model specified in this section is implemented as an optional tally extension that is **disabled by default** and can be activated only through an on-chain governance proposal. Until activation, all tallies described below reduce to the duration-weighted model.

When activated, QDRW replaces linear stake-weighted voting with a quadratic-reputation-weighted system designed to resist plutocratic capture. The voting power of a participant with staked amount s, locked xQORE amount x, and reputation score r is:

$$VP(s, x, r) = \sqrt{s + 2x} \cdot M(r) \tag{10.40}$$

where $M(r)$ is the reputation multiplier from Equation 10.11.

The three components serve distinct purposes:

- **Square root dampening ($\sqrt{\cdot}$):** Prevents linear scaling of voting power with capital. A participant who stakes 100x more than another gains only $\sqrt{100} = 10$x the voting power. This is the primary anti-whale mechanism.
- **xQORE boost ($2x$):** Locked governance tokens (xQORE) contribute double their value to the voting power calculation. This rewards participants who commit to long-term governance participation by locking QOR through the xQORE mechanism (1:1 lock ratio, graduated exit penalties).
- **Reputation multiplier ($M(r)$):** Maps $r \in [0, 1]$ to $[0.5, 2.0]$. Validators and delegators with strong track records receive up to 2x voting weight, while those with poor histories are dampened to 0.5x.

10.8.2 Quadratic Dampening Analysis

The quadratic dampening effect is quantified by the ratio of voting power to stake:

$$\frac{VP(s, 0, 0.5)}{s} = \frac{\sqrt{s} \cdot M(0.5)}{s} = \frac{M(0.5)}{\sqrt{s}} \tag{10.41}$$

This ratio decreases as $O(1/\sqrt{s})$, meaning larger stakeholders have a diminishing marginal return on their voting influence. Concrete examples:

Table 10.7: QDRW Dampening Effect (xQORE = 0, $r = 0.5$)

Stake (QOR)	VP (QDRW)	VP (Linear)	Dampening
100	12.5	100	8.0x
10,000	125.0	10,000	80.0x
1,000,000	1,250.0	1,000,000	800.0x
100,000,000	12,500.0	100,000,000	8,000.0x

10.8.3 xQORE Contribution

The xQORE mechanism allows participants to lock QOR tokens (1:1 ratio) to gain enhanced governance weight. The locking contract enforces graduated exit penalties:

Table 10.8: xQORE Exit Penalty Schedule

Lock Duration	Penalty	Destination
< 30 days	50%	Redistributed to remaining xQORE holders (PvP rebase)
30–90 days	35%	Redistributed to remaining xQORE holders
90–180 days	15%	Redistributed to remaining xQORE holders
> 180 days	0%	Full withdrawal

The xQORE multiplier of 2 means that a participant who locks 1,000 QOR as xQORE gains the same governance contribution as staking 2,000 QOR directly. The sensitivity of voting power to xQORE, relative to plain stake, is:

$$\frac{\partial VP}{\partial x} = \frac{2}{2\sqrt{s + 2x \cdot M(r) = \frac{M(r)}{\sqrt{s+2x}}}} \tag{10.42}$$

$$\frac{\partial VP}{\partial s} = \frac{1}{2\sqrt{s + 2x \cdot M(r) = \frac{M(r)}{2\sqrt{s+2x}}}} \tag{10.43}$$

The ratio $\frac{\partial VP/\partial x}{\partial VP/\partial s} = 2$, confirming that xQORE is exactly twice as efficient as plain stake for governance influence. This creates a strong incentive to lock tokens, reducing circulating supply and signalling long-term commitment.

10.8.4 Whale Resistance

Theorem 11 (QDRW Whale Resistance)**.** *Under QDRW, a participant holding fraction α of total stake controls at most $\sqrt{\alpha}$ fraction of total voting power. For $\alpha = 0.51$ (majority stake), the maximum voting power share is $\sqrt{0.51} \approx 0.714$, which is below the $2/3$ supermajority threshold required for governance proposals.*

Proof. Let total stake be S_{total} and the whale's stake be $s_w = \alpha \cdot S_{\text{total}}$. The whale's voting power is $VP_w = \sqrt{s_w} \cdot M(r_w)$. The remaining participants have total voting power $VP_{\text{rest}} = \sum_{i \neq w} \sqrt{s_i} \cdot M(r_i)$. By the Cauchy-Schwarz inequality:

$$\sum_{i \neq w} \sqrt{s_i} \geq \sqrt{(1 - \alpha) \cdot S_{\text{total}}}$$

The whale's share of voting power is bounded by:

$$\frac{VP_w}{VP_w + VP_{\text{rest}}} \leq \frac{\sqrt{\alpha} \cdot M_{\max}}{\sqrt{\alpha} \cdot M_{\max} + \sqrt{1 - \alpha} \cdot M_{\min}}$$

For $M_{\max}/M_{\min} = 2.0/0.5 = 4$, even with maximum reputation advantage, a 51% stakeholder controls at most $\frac{4\sqrt{0.51}}{4\sqrt{0.51}+\sqrt{0.49}} \approx 0.80$ of voting power. Without reputation advantage ($M_w = M_{\text{avg}}$), the share is $\sqrt{0.51} \approx 0.714 < 2/3$. □

10.8.5 Gini Coefficient Under QDRW

The Gini coefficient of voting power under QDRW, compared to linear voting, is:

$$G_{\text{QDRW}} \approx \frac{G_{\text{linear}}}{2} + O\left(\frac{\text{Var}(M(r))}{n}\right) \tag{10.44}$$

For highly concentrated stake distributions ($G_{\text{linear}} \to 1$), QDRW approximately halves the inequality. This is a direct consequence of the square-root transformation compressing the upper tail of the distribution.

```
# Query voting power for a participant
qorechaind query qca voting-power \
  --address qor1abc...def \
  --output json

# Example response:
# {
#   "address": "qor1abc...def",
#   "staked": "500000000000",
#   "xqore_locked": "200000000000",
#   "reputation": "0.87",
#   "voting_power": "11847.23",
#   "components": {
#     "sqrt_input": "900000000000",
#     "sqrt_result": "948683.30",
#     "reputation_multiplier": "1.82",
#     "final_vp": "1726607.61"
#   }
# }

# Submit a governance proposal
qorechaind tx gov submit-proposal \
  --title "Adjust RL Agent Mode to Autonomous" \
  --description "Proposal to enable full RL-driven consensus tuning" \
  --type text \
  --from qor1abc...def \
  --chain-id qorechain-diana

# Query governance tally with QDRW weights
qorechaind query gov tally 1 --output json
```

10.9 Formal Consensus Security Analysis

10.9.1 Safety Under Byzantine Faults

Byzantine fault tolerance depends on the fraction of malicious stake remaining below the safety threshold. Under CPoS, this threshold is calibrated to ensure that the economic cost of a successful attack exceeds the potential gain from the attack. In QoreChain, validators must stake QOR tokens and face slashing penalties that con-

sume a significant fraction of their stake if they misbehave. This creates an economic disincentive whose magnitude depends on the ratio of attack payoff to slashing penalty. Combined with the mathematical 1/3 threshold, this two-layer defense (economic and cryptographic) makes attacks economically irrational for rational validators. The safety guarantee extends to cross-chain operations: bridge validators represent a distinct pool and must satisfy the same Byzantine threshold across their cohort, ensuring that malicious bridge validators cannot unilaterally sign fraudulent attestations.

Theorem 12 (CPoS Safety)**.** *The QoreChain consensus is safe (no conflicting blocks are finalised at the same height) as long as the Byzantine validator stake fraction satisfies $f < n/3$, where n is the total voting power.*

Proof sketch. Block finalisation requires $> 2/3$ of voting power to agree. For two conflicting blocks B_1 and B_2 to be finalised at height h, both must receive $> 2/3$ agreement. This requires a total agreement of $> 4/3$, which is impossible since total voting power is 1. Therefore, at least one set of agreeing validators must overlap between B_1 and B_2, meaning some honest validator voted for both, which contradicts the assumption that honest validators vote for at most one block per height. □

10.9.2 Liveness Under CPoS

Theorem 13 (CPoS Liveness)**.** *If $> 2/3$ of voting power is held by honest, reachable validators, the chain produces new finalised blocks indefinitely.*

The triple-pool architecture does not affect the liveness guarantee because pool weights only determine *which* validator proposes, not *whether* a proposal occurs. As long as at least one pool contains an honest validator with non-zero selection probability, proposals continue. Since all honest validators have positive selection probability under CPoS (Equation 10.4), liveness holds whenever the honest majority condition is satisfied.

More formally, liveness follows from two properties: (1) the proposer selection mechanism guarantees that in each block height, some validator in the union of all pools will be selected with positive probability, and (2) this probability remains bounded away from zero as long as the honest majority holds. Let $f < 1/3$ denote the fraction of Byzantine validators. The minimum proposal probability for an honest validator in any pool is bounded below by a constant depending on pool weights and the consensus stake distribution. Therefore, within any window of $O(1/f)$ blocks, the chain is guaranteed to produce at least one block proposed and finalised by an honest validator.

The liveness guarantee extends to the fee market: even if transaction fees spike, honest validators will continue proposing and finalising blocks (though potentially with lower transaction density). The system enters adaptive congestion control (described in Section **??**) when the mempool exceeds threshold size, ensuring that liveness is never compromised by load.

Furthermore, liveness is maintained under the RL agent's Autonomous mode because the circuit breaker (Theorem **??**) guarantees that total reward degradation is bounded. No configuration of RL weights can eliminate the proposal opportunity for honest validators; they can only be temporarily deprioritised. Thus, liveness holds as long as the honest majority and network connectivity assumptions hold.

10.9.3 RL Safety Bound

The RL agent introduces a novel attack surface: a compromised governance could set the agent to Autonomous mode with adversarial policy weights, degrading chain performance. The circuit breaker bounds this risk:

Theorem 14 (RL Damage Bound)**.** *Under circuit breaker protection with window W and threshold θ_{CB}, the maximum number of degraded blocks before automatic recovery is:*

$$B_{degraded} \leq W \cdot (1 - \theta_{CB}) + W = W \cdot (2 - \theta_{CB})$$

For $W = 50$ and $\theta_{CB} = 0.5$: $B_{degraded} \leq 75$ blocks.

Proof. In the worst case, the circuit breaker requires $W \cdot (1 - \theta_{\text{CB}})$ degraded blocks within the window to trigger. After triggering, the recovery period is W blocks with default parameters. Total degraded blocks: at most $W \cdot (1 - \theta_{\text{CB}})$ (pre-trigger) + 0 (post-trigger, since defaults are safe). Adding the recovery period as a conservative upper bound gives $W \cdot (2 - \theta_{\text{CB}})$. □

10.9.4 Game-Theoretic Equilibrium

The combined incentive structure (bonding curve rewards + progressive slashing + QDRW governance) creates a Nash equilibrium at honest behaviour:

$$\mathbb{E}[U_{\text{honest}}] = R(v,t) - 0 > \mathbb{E}[U_{\text{dishonest}}] = R(v,t) + g \cdot P_{\text{undetected}} - s \cdot (1 - P_{\text{undetected}}) \tag{10.45}$$

For this to hold, the required slashing penalty is:

$$s > \frac{g \cdot P_{\text{undetected}}}{1 - P_{\text{undetected}}} \tag{10.46}$$

With AI-powered anomaly detection ($P_{\text{undetected}} < 0.005$) and progressive slashing (s escalates with repeated attempts), the condition is satisfied for any realistic attack gain g after at most two attempts:

$$s(k = 2) = 0.01 \cdot 1.5^2 \cdot \text{severity} = 0.0225 \cdot \text{severity} \tag{10.47}$$

For double-sign (severity = 5.0): $s(k = 2) = 11.25\%$, exceeding the Nash condition for gains $g < 11.25\%/0.005 \approx 2250\%$ of validator stake, an unrealistically high attack gain.

10.9.5 Consensus Security Summary

Table 10.9: Consensus Security Properties

Property	**Mechanism**	**Guarantee**
Safety (no forks)	BFT finality	Holds for $f < n/3$
Liveness (progress)	CPoS proposer selection	Holds for honest $> 2/3$
Decentralisation	Triple-pool CPoS	$G_{\text{CPoS}} < G_{\text{stake}}$
Adaptive performance	RL agent	Parameter tuning within safe bounds
RL safety	Circuit breaker	≤ 75 degraded blocks worst case
Whale resistance	QDRW governance (optional, when activated)	51% stake $\rightarrow \leq 71.4\%$ voting power
Honest equilibrium	Bonding + slashing	Nash equilibrium at honest behaviour
Quantum resistance	ML-DSA-87	$\geq 2^{128}$ quantum gate security

```
# Query overall consensus security metrics
qorechaind query qca security-summary --output json

# Example response:
# {
#   "byzantine_threshold": "0.333",
#   "current_honest_fraction": "0.97",
#   "safety_margin": "0.637",
#   "gini_coefficient": {
#     "stake_weighted": "0.72",
#     "cpos_weighted": "0.51"
#   },
#   "nakamoto_coefficient": {
#     "stake_weighted": 4,
#     "cpos_weighted": 7
#   },
#   "rl_agent": {
#     "mode": "conservative",
#     "circuit_breaker": "inactive",
#     "max_degraded_blocks": 75
#   },
#   "qdrw": {
#     "enabled": true,
#     "largest_vp_share": "0.089"
#   }
# }
```

Chapter 11

QOR Token Economics

The QOR token is the native unit of account, fee medium, governance instrument, and staking asset of the QoreChain network. This chapter presents the formal economic model governing token supply, emission, burn mechanics, fee distribution, and governance-boosted staking, together with the mathematical analysis that demonstrates long-term economic sustainability.

11.1 Token Overview and Supply Model

11.1.1 Denomination

QOR is the human-readable display denomination. The on-chain base denomination is `uqor`, with the conversion:

$$1 \text{ QOR} = 10^6 \text{ uqor} \tag{11.1}$$

All on-chain arithmetic operates in `uqor` to avoid floating-point precision issues. The Bech32 address prefix is `qor` for accounts and `qorvaloper` for validators.

The choice of 10^6 (one million) subunits per QOR provides sufficient granularity for both small micropayments and large institutional transfers. Balances and transaction amounts are stored as unsigned 256-bit integers in `uqor`, allowing total supply up to approximately $2^{256}/10^6$ QOR. This avoids floating-point arithmetic entirely, eliminating precision errors that plague other blockchains.

The `uqor` denomination is used throughout the codebase: in genesis parameters, transaction fees, staking amounts, governance proposals, and all state machine operations. User-facing interfaces (wallets, explorers) automatically convert `uqor` to QOR for display. The notation $\text{QOR}(x)$ denotes the display value of x uqor, computed as $x/10^6$. For cryptographic verification and state machine reasoning, all analysis is conducted in `uqor`.

11.1.2 Total Supply

The genesis total supply is fixed at:

$$S_0 = 4{,}500{,}000{,}000 \text{ QOR} = 4.5 \times 10^{15} \text{ uqor} \tag{11.2}$$

The effective circulating supply at time t is governed by the emission-burn differential:

$$S_{\text{circ}}(t) = S_0 + \int_0^t \epsilon(\tau)\, d\tau - \int_0^t \beta(\tau)\, d\tau - L(t) \tag{11.3}$$

where $\epsilon(\tau)$ is the instantaneous emission rate, $\beta(\tau)$ is the instantaneous burn rate, and $L(t)$ is the total locked supply (staked + xQORE + rollup bonds + bridge escrow).

11.1.3 Supply Decomposition

At any time t, the total token supply is partitioned into disjoint categories:

$$S_{\text{total}}(t) = S_{\text{circ}}(t) + S_{\text{staked}}(t) + S_{\text{xQORE}}(t) + S_{\text{treasury}}(t) + S_{\text{escrow}}(t) + S_{\text{burned}}(t) \tag{11.4}$$

where:

- $S_{\text{circ}}(t)$: freely transferable tokens
- $S_{\text{staked}}(t)$: tokens delegated to validators (subject to unbonding period)
- $S_{\text{xQORE}}(t)$: tokens locked in xQORE governance contracts
- $S_{\text{treasury}}(t)$: protocol treasury holdings
- $S_{\text{escrow}}(t)$: tokens held in bridge escrow and rollup bonds
- $S_{\text{burned}}(t)$: permanently removed from supply

The burned supply is monotonically non-decreasing: $S_{\text{burned}}(t_2) \geq S_{\text{burned}}(t_1)$ for $t_2 > t_1$, since burned tokens cannot be recovered.

```
# Query total supply and decomposition
qorechaind query bank total --denom uqor

# Query tokenomics overview
qorechaind query tokenomics overview --output json

# Example response:
# {
#   "total_supply": "4500000000000000",
#   "circulating": "2835000000000000",
#   "staked": "1125000000000000",
#   "xqore_locked": "225000000000000",
#   "treasury": "270000000000000",
#   "escrow": "13500000000000",
#   "burned": "31500000000000",
#   "denom": "uqor"
# }
```

11.2 Epoch-Based Emission Schedule

11.2.1 Annual Emission Rate

QoreChain implements a decaying emission schedule that provides strong early incentives while converging toward minimal long-term inflation. The annual emission rate ϵ_y is a step function of the network year y:

$$\epsilon(y) = \begin{cases} 0.175 & y = 1 \\ 0.110 & y = 2 \\ 0.070 & y \in \{3, 4\} \\ 0.020 & y \geq 5 \end{cases} \tag{11.5}$$

The emission rate can also be expressed as a continuous approximation using an exponential decay envelope:

$$\hat{\epsilon}(y) = \epsilon_\infty + (\epsilon_1 - \epsilon_\infty) \cdot e^{-\xi(y-1)}, \quad \xi = \frac{\ln(\epsilon_1/\epsilon_\infty)}{y_{\text{converge}}} \tag{11.6}$$

where $\epsilon_1 = 0.175$, $\epsilon_\infty = 0.02$, and $y_{\text{converge}} \approx 5$ is the convergence year. This gives $\xi \approx 0.438$.

11.2.2 Per-Epoch Minting

The network operates in discrete epochs of configurable length E_{len} blocks (default: 100). Given B_{year} blocks per year, the number of epochs per year is:

$$N_{\text{epochs}} = \frac{B_{\text{year}}}{E_{\text{len}}} \tag{11.7}$$

The mint amount per epoch in year y is:

$$M_{\text{epoch}}(y) = \frac{\epsilon(y) \cdot S_{\text{total}}}{N_{\text{epochs}}} \tag{11.8}$$

For a target block time of 5 seconds: $B_{\text{year}} = \frac{365.25 \times 24 \times 3600}{5} = 6{,}311{,}520$ blocks/year, giving $N_{\text{epochs}} = 63{,}115.2$ epochs/year. In Year 1:

$$M_{\text{epoch}}^{(1)} = \frac{0.175 \times 4.5 \times 10^9}{63{,}115} \approx 12{,}478 \text{ QOR per} epoch \tag{11.9}$$

11.2.3 Cumulative Supply Trajectory

The cumulative minted supply after Y full years is:

$$S_{\text{minted}}(Y) = S_0 \cdot \sum_{y=1}^{Y} \epsilon(y) \tag{11.10}$$

Table 11.1: Cumulative Emission Schedule

Year	Rate	Annual Emission (QOR)	Cumulative Minted (QOR)
1	17.5%	787,500,000	787,500,000
2	11.0%	495,000,000	1,282,500,000
3	7.0%	315,000,000	1,597,500,000
4	7.0%	315,000,000	1,912,500,000
5	2.0%	90,000,000	2,002,500,000
10	2.0%	90,000,000	2,452,500,000

11.2.4 Emission Convergence

The long-term annual emission converges to a fixed rate:

$$\lim_{y \to \infty} \epsilon(y) \cdot S_0 = 0.02 \times 4.5 \times 10^9 = 90{,}000{,}000 \text{ QOR}/year \tag{11.11}$$

As a fraction of cumulative total supply (genesis + all minted), the effective inflation rate decreases monotonically:

$$\epsilon_{\text{eff}}(Y) = \frac{\epsilon(Y) \cdot S_0}{S_0 + S_{\text{minted}}(Y)} < \epsilon(Y) \tag{11.12}$$

For $Y = 10$: $\epsilon_{\text{eff}}(10) = \frac{90{,}000{,}000}{4{,}500{,}000{,}000 + 2{,}452{,}500{,}000} \approx 1.29\%$, meaning that even before accounting for burns, the effective inflation rate falls below the nominal 2% by Year 10.

```
# Query current inflation rate and epoch info
qorechaind query inflation rate --output json

# Example response:
# {
#   "annual_rate": "0.175",
#   "effective_rate": "0.163",
#   "current_epoch": 14523,
#   "current_year": 1,
#   "epoch_length": 100,
#   "mint_per_epoch": "12478000000",
#   "total_minted": "181234567000000",
#   "denom": "uqor"
# }

# Query emission schedule
qorechaind query inflation schedule --output json
```

11.3 Deflationary Burn Engine

11.3.1 Ten-Channel Burn Architecture

QoreChain implements a central burn accounting system fed by ten distinct burn channels. Each channel captures a different source of economic activity, ensuring that burn

pressure scales with network usage across all dimensions.

Table 11.2: Burn Channel Taxonomy

ID	Channel	Source
1	`tx_fee`	30% of all transaction fees (via End-Blocker distribution)
2	`governance_penalty`	Deposits forfeited from failed governance proposals
3	`slashing_burn`	Portion of validator slashing penalties
4	`bridge_fee`	Bridge transfer protocol fees
5	`spam_deterrent`	Failed and rejected transaction fees
6	`epoch_excess`	Excess emissions beyond target in an epoch
7	`manual_burn`	Governance-initiated discretionary burns
8	`contract_callback`	Smart contract deployment fees
9	`cross_vm_fee`	Cross-VM messaging operation fees
10	`rollup_create`	1% of rollup creation bond

11.3.2 Aggregate Burn Rate

The total burn rate at time t is the sum across all channels:

$$\beta(t) = \sum_{k=1}^{10} \beta_k(t) \tag{11.13}$$

The dominant channel is `tx_fee`, which accounts for the largest share of burn under normal conditions. The transaction fee burn rate is:

$$\beta_1(t) = 0.30 \cdot \text{TxVolume}(t) \cdot \overline{f}(t) \tag{11.14}$$

where $\text{TxVolume}(t)$ is the number of transactions per unit time and $\overline{f}(t)$ is the average fee per transaction.

11.3.3 Burn Rate Modelling

The burn rate is inherently tied to network usage. We model the aggregate burn as a function of network utilisation $u(t) \in [0, 1]$ (fraction of gas capacity consumed):

$$\beta(t) = \beta_{\text{base}} + \beta_{\text{scale}} \cdot u(t)^{\gamma_b} \tag{11.15}$$

where β_{base} captures minimum burns from governance penalties, slashing, and bridge fees, β_{scale} is the maximum additional burn at full utilisation, and $\gamma_b > 1$ is a convexity parameter reflecting the EIP-1559-style fee escalation under congestion.

The per-block expected burn under a Poisson transaction arrival model with rate λ_{tx} is:

$$\mathbb{E}[\beta_{\text{block}}] = 0.30 \cdot \lambda_{\text{tx}} \cdot \bar{f} + \beta_{\text{bridge}} \cdot \lambda_{\text{bridge}} + \beta_{\text{rollup}} \cdot \lambda_{\text{rollup}} + \beta_{\text{residual}} \tag{11.16}$$

where λ_{bridge} and λ_{rollup} are the arrival rates for bridge and rollup creation transactions respectively, and β_{residual} captures the remaining channels.

11.3.4 Burn Statistics and Tracking

The burn module maintains per-channel cumulative statistics:

$$B_k(t) = \int_0^t \beta_k(\tau)\, d\tau, \quad k \in \{1, 2, \ldots, 10\} \tag{11.17}$$

The total burned supply is:

$$S_{\text{burned}}(t) = \sum_{k=1}^{10} B_k(t) \tag{11.18}$$

```
# Query burn statistics
qorechaind query burn stats --output json

# Example response:
# {
#   "total_burned": "31500000000000",
#   "per_channel": {
#     "tx_fee": "22050000000000",
#     "governance_penalty": "450000000000",
#     "slashing_burn": "315000000000",
#     "bridge_fee": "4725000000000",
#     "spam_deterrent": "900000000000",
#     "epoch_excess": "135000000000",
#     "manual_burn": "0",
#     "contract_callback": "1575000000000",
#     "cross_vm_fee": "675000000000",
#     "rollup_create": "675000000000"
#   },
#   "last_burn_height": 142857,
#   "denom": "uqor"
# }
```

11.4 Fee Distribution Model

11.4.1 Five-Way Split

Every block, the collected transaction fees are distributed across five recipients through the EndBlocker mechanism:

$$F_{\text{block}} = \underbrace{0.37 \cdot F}_{\text{Validators}} + \underbrace{0.30 \cdot F}_{\text{Burn}} + \underbrace{0.20 \cdot F}_{\text{Treasury}} + \underbrace{0.10 \cdot F}_{\text{Stakers}} + \underbrace{0.03 \cdot F}_{\text{Light Nodes}} \tag{11.19}$$

where F is the total fees collected in the block. The distribution preserves the accounting identity: all shares sum to exactly 1.0.

The five-way split balances multiple network incentives. Validators receive 37% to compensate block production and consensus participation. Burn at 30% reduces circulating supply and counters inflation. Treasury at 20% funds ecosystem development through governance. Stakers receive 10% of fees in addition to emission rewards,

incentivising long-term xQOR locking. Light nodes receive 3% as compensation for maintaining full state archives and serving state queries.

Each recipient receives their allocation via the EndBlocker fee distribution at block finality. Validator rewards are distributed proportionally according to their governance voting power in the current epoch (duration-weighted at genesis; QDRW-weighted if that extension is activated). Treasury funds accumulate in a community pool governed through Tier 2 proposals. Staker fees are automatically added to the global reward pool and claimed by delegators in the next epoch. Light node fees require on-chain registration and proof of archive commitment; nodes can register through `RegisterLightNode` transactions.

11.4.2 Recipient Revenue Models

Validator Revenue

Validators receive revenue from three sources: fee distribution, bonding curve rewards, and epoch emissions. The total validator revenue per epoch is:

$$\text{Rev}_v(\text{epoch}) = \underbrace{0.37 \cdot F_{\text{epoch}}}_{\text{fee share}} + \underbrace{R(v,t)}_{\text{bonding curve}} + \underbrace{\frac{s_v}{\sum_j s_j} \cdot M_{\text{epoch}}(y) \cdot \alpha_{\text{val}}}_{\text{emission share}} \tag{11.20}$$

where α_{val} is the fraction of epoch emissions allocated to validators.

Treasury Growth

The treasury balance evolves as:

$$\frac{dS_{\text{treasury}}}{dt} = 0.20 \cdot F(t) - G(t) \tag{11.21}$$

where $G(t)$ is the governance-approved spending rate. The treasury reaches a steady state when spending matches inflow: $G^* = 0.20 \cdot F^*$.

Staker Revenue

Passive stakers (delegators) receive 10% of block fees proportional to their stake:

$$\text{Rev}_{\text{staker}}(i) = 0.10 \cdot F_{\text{epoch}} \cdot \frac{d_i}{\sum_j d_j} \tag{11.22}$$

where d_i is the delegated stake of participant i.

Light Node Revenue

Light node operators share 3% of block fees, distributed proportionally to their stake-weighted uptime (as detailed in Chapter 8):

$$\text{Rev}_{\text{LN}}(i) = 0.03 \cdot F_{\text{epoch}} \cdot \frac{w_i \cdot u_i}{\sum_j w_j \cdot u_j} \tag{11.23}$$

where w_i is the delegated stake and u_i is the uptime factor of light node i.

11.4.3 Fee Sensitivity Analysis

The marginal impact of a 1% increase in transaction volume on each recipient's revenue is:

$$\frac{\partial \text{Rev}_r}{\partial \text{TxVol}} = \alpha_r \cdot \frac{\partial F}{\partial \text{TxVol} = \alpha_r \cdot \overline{f}} \tag{11.24}$$

where α_r is the share for recipient r and $\overline{f}$ is the average fee. This linear relationship means all recipients benefit equally (in percentage terms) from increased network activity.

The annual fee revenue, under the assumption of average utilisation $\bar{u}$ and average fee $\bar{f}$, is:

$$F_{\text{annual}} = B_{\text{year}} \cdot \bar{u} \cdot \text{GasLimit} \cdot \bar{p}_{\text{gas}} \tag{11.25}$$

```
# Query fee distribution parameters
qorechaind query burn fee-distribution --output json

# Query accumulated fees for current epoch
qorechaind query burn epoch-fees --output json

# Example response:
# {
#   "epoch": 14523,
#   "total_fees_collected": "4523870000",
#   "distribution": {
#     "validators": "1673832900",
#     "burned": "1357161000",
#     "treasury": "904774000",
#     "stakers": "452387000",
#     "light_nodes": "135716100"
#   },
#   "denom": "uqor"
# }
```

11.5 xQORE Governance-Boosted Staking

11.5.1 Lock-Mint Mechanism

The xQORE system allows participants to lock QOR tokens and receive xQORE governance tokens at a 1:1 ratio. Locking is irreversible for a minimum period; early exits incur graduated penalties. Formally:

$$\text{Lock}(q) \to \text{Mint}(x = q), \quad q \in \mathbb{Z}^+, \; q \leq \text{Balance}(\text{sender}) \tag{11.26}$$

The locked QOR is held in the xQORE module account and is non-transferable until unlocked. At genesis, xQORE positions receive duration-based governance multipliers (1.0x to 2.0x depending on lock length); if the optional QDRW tally (Section 10.8)

is activated by governance, xQORE additionally counts at double weight inside the QDRW formula.

11.5.2 Graduated Exit Penalties

The exit penalty P_{exit} is a decreasing step function of lock duration Δt:

$$P_{\text{exit}}(\Delta t) = \begin{cases} 0.50 & \Delta t < 30 \text{ days} \\ 0.35 & 30 \leq \Delta t < 90 \text{ days} \\ 0.15 & 90 \leq \Delta t < 180 \text{ days} \\ 0.00 & \Delta t \geq 180 \text{ days} \end{cases} \tag{11.27}$$

The penalty can also be expressed as a continuous approximation:

$$\hat{P}_{\text{exit}}(\Delta t) = 0.50 \cdot e^{-\frac{\Delta t}{\tau_p, \quad \tau_p \approx 78 \text{ days}}} \tag{11.28}$$

where τ_p is fitted to match the step function's average behaviour. The net amount received upon early exit is:

$$q_{\text{received}} = x \cdot (1 - P_{\text{exit}}(\Delta t)) \tag{11.29}$$

11.5.3 PvP Rebase Redistribution

The penalty amount is not burned; it is redistributed to all remaining xQORE holders proportionally. This creates a player-vs-player (PvP) rebase mechanism where early exits directly benefit long-term holders.

When participant j exits with penalty $P_j \cdot x_j$, each remaining holder i receives:

$$\Delta x_i = P_j \cdot x_j \cdot \frac{x_i}{\sum_{k \neq j} x_k} \tag{11.30}$$

The effective APY from PvP rebase alone, assuming an average exit rate of λ_{exit} per epoch with average penalty $\bar{P}$, is:

$$\text{APY}_{\text{PvP}} = \lambda_{\text{exit}} \cdot \bar{P} \cdot N_{\text{epochs}}/year \cdot \frac{\bar{x}_{\text{exiting}}}{\sum_k x_k - \bar{x}_{\text{exiting}}} \tag{11.31}$$

11.5.4 Game-Theoretic Lock Duration

The optimal lock duration depends on the participant's time preference discount rate δ and expected PvP rebase yield. A rational participant with discount rate δ computes the net present value of holding xQORE for duration T:

$$\text{NPV}(T) = -q + \sum_{t=1}^{T} \frac{r_{\text{PvP}}(t) + r_{\text{gov}}(t)}{(1+\delta)^t} + \frac{q \cdot (1 - P_{\text{exit}}(T))}{(1+\delta)^T} \tag{11.32}$$

where $r_{\text{PvP}}(t)$ is the PvP rebase reward at epoch t and $r_{\text{gov}}(t)$ is the additional governance participation reward. The optimal lock duration T^* satisfies:

$$T^* = \arg\max_T \text{NPV}(T) \tag{11.33}$$

For participants with low discount rates ($\delta < 0.1$), $T^* > 180$ days (full lock period), as the cumulative PvP rebase and governance rewards exceed the time cost of locking. For high discount rates ($\delta > 0.5$), $T^* = 0$ (no locking), as the immediate opportunity cost exceeds the expected rewards.

Theorem 15 (xQORE Participation Incentive)**.** *Under steady-state conditions with constant exit rate $\lambda_{exit} > 0$ and average penalty $\bar{P} > 0$, the expected return for a holder who completes the full 180-day lock period strictly exceeds the return for a non-participant:*

$$\mathbb{E}[Return_{180}] > \mathbb{E}[Return_{none}]$$

This holds because the PvP rebase is a zero-sum transfer from early exiters to long-term holders, and the full-lock participant pays zero penalty.

```
# Lock QOR to mint xQORE
qorechaind tx xqore lock 1000000000uqor \
  --from qor1abc...def \
  --chain-id qorechain-diana

# Query xQORE position
qorechaind query xqore position qor1abc...def --output json

# Example response:
# {
#   "address": "qor1abc...def",
#   "locked_qor": "1000000000",
#   "xqore_balance": "1023456789",
#   "lock_height": 130000,
#   "lock_time": "2026-02-15T12:00:00Z",
#   "current_penalty": "0.15",
#   "days_locked": 34,
#   "accumulated_rebase": "23456789"
# }

# Estimate exit penalty
qorechaind query xqore exit-estimate \
  --address qor1abc...def \
  --output json

# Unlock xQORE (subject to penalty if < 180 days)
qorechaind tx xqore unlock 500000000uxqore \
  --from qor1abc...def \
  --chain-id qorechain-diana
```

11.6 Economic Equilibrium Analysis

11.6.1 Net Emission Function

The net change in circulating supply per unit time is the emission-burn differential:

$$\frac{dS_{\text{circ}}}{dt} = \epsilon(t) - \beta(t) - \frac{dL}{dt} \tag{11.34}$$

where $\frac{dL}{dt}$ is the net locking rate (staking + xQORE locking minus unstaking minus unlocking). The network enters a deflationary regime when:

$$\beta(t) > \epsilon(t) + \frac{dL}{dt} \quad \Longleftrightarrow \quad \text{Net supply change} < 0 \tag{11.35}$$

11.6.2 Deflationary Threshold

We define the deflationary threshold u^* as the minimum network utilisation at which burn exceeds emission. Setting $\beta(u^*) = \epsilon(y)$ and using the burn-utilisation model (Equation 11.15):

$$\beta_{\text{base}} + \beta_{\text{scale}} \cdot (u^*)^{\gamma_b} = \epsilon(y) \cdot S_0 / B_{\text{year}} \tag{11.36}$$

Solving for u^*:

$$u^* = \left(\frac{\epsilon(y) \cdot S_0 / B_{\text{year}} - \beta_{\text{base}}}{\beta_{\text{scale}}} \right)^{1/\gamma_b} \tag{11.37}$$

As y increases, $\epsilon(y)$ decreases, making u^* lower. By Year 5, with $\epsilon(5) = 0.02$, the deflationary threshold is substantially lower than in Year 1 ($\epsilon(1) = 0.175$), meaning moderate network activity is sufficient to achieve net deflation.

11.6.3 Steady-State Supply

In the long run ($y \geq 5$), the emission rate is constant at 2%. The steady-state supply S^* is reached when annual emissions equal annual burns:

$$0.02 \cdot S_0 = \beta_{\text{annual}}(S^*, u^*, \bar{f}^*) \tag{11.38}$$

If the burn rate grows with network usage (which increases with adoption), the steady-state circulating supply satisfies:

$$S^*_{\text{circ}} = S_0 + S^{\infty}_{\text{minted}} - S^{\infty}_{\text{burned}} - L^* \tag{11.39}$$

where $S^{\infty}_{\text{minted}}$ is the cumulative minted supply (convergent series if burn exceeds emission) and L^* is the steady-state locked amount.

11.6.4 Token Velocity Model

The Fisher equation of exchange adapted for a blockchain economy:

$$M \cdot V = P \cdot Q \tag{11.40}$$

where M is the token monetary base (S_{circ}), V is velocity (average number of times each token changes hands per year), P is the price level of on-chain services, and Q is the quantity of on-chain economic activity.

The xQORE mechanism directly reduces velocity by locking tokens:

$$V_{\text{effective}} = \frac{V_{\text{base}} \cdot S_{\text{circ}}}{S_{\text{circ}} + S_{\text{xQORE}}} < V_{\text{base}} \tag{11.41}$$

since locked xQORE tokens have zero velocity. The greater the xQORE adoption, the lower the effective velocity, which (holding $P \cdot Q$ constant) implies a higher equilibrium value per token.

11.6.5 Supply Shock Analysis

Large-scale burns or lock-ups create supply shocks. The supply elasticity of price, under constant demand, is:

$$\eta_S = \frac{\Delta P/P}{\Delta S/S} = -\frac{1}{1-\sigma} \tag{11.42}$$

where $\sigma \in [0, 1)$ is the demand elasticity. For relatively inelastic demand ($\sigma \to 0$, as expected for a utility token), $\eta_S \to -1$: a 1% reduction in supply leads to approximately 1% price increase. The combined effect of burn + xQORE locking creates sustained supply-side pressure.

Supply shock scenarios include large coordinated xQORE lock-ins (e.g., institutional staking campaigns) or accelerated burn through high-volume transaction periods. Under the elasticity model, a 10% reduction in circulating supply (through burn or xQORE adoption) implies approximately 10% price increase in the absence of compensatory demand shifts. This provides a powerful deflationary mechanism but also introduces tail risk: if transaction demand drops sharply while burn continues, price volatility can spike.

Mitigation strategies include governance-controllable burn rates and elastic fee markets. The burn coefficient can be reduced from the default 30% down to 20% (Tier 2 governance) if economic conditions suggest supply is becoming too scarce relative to demand. Additionally, the fee market adjusts gas prices dynamically; high demand increases fees, which increases burn but signals users to reduce non-critical transactions, naturally dampening demand spikes.

Long-term supply trajectory analysis employs Monte Carlo simulation under random transaction volume. The median predicted 5-year supply, under baseline assumptions (3-year emission decay plus 30% burn), is approximately 25% below the initial 1 billion QOR, with 95% confidence interval ranging from 18% to 32% reduction. This implies substantial long-term deflationary pressure, supporting QOR value per unit.

11.6.6 Monte Carlo Supply Simulation

The long-term supply trajectory can be simulated under stochastic transaction volume. Let $\lambda_{\text{tx}}(t) \sim \text{LogNormal}(\mu_\lambda, \sigma_\lambda^2)$ be the random daily transaction count. The supply at year Y after N simulation runs is:

$$\hat{S}_{\text{circ}}(Y) = \frac{1}{N} \sum_{n=1}^{N} \left[S_0 + \sum_{y=1}^{Y} \epsilon(y) \cdot S_0 - \sum_{t=1}^{Y \cdot 365} \beta^{(n)}(t) - L^{(n)}(Y) \right] \tag{11.43}$$

The 95% confidence interval provides a range for the expected circulating supply:

$$\text{CI}_{95\%} = \hat{S}_{\text{circ}}(Y) \pm 1.96 \cdot \frac{s_N}{\sqrt{N}} \tag{11.44}$$

where s_N is the sample standard deviation across simulation runs.

```
# Query comprehensive tokenomics metrics
qorechaind query tokenomics equilibrium --output json

# Example response:
# {
#   "current_year": 1,
#   "emission_rate": "0.175",
#   "effective_inflation": "0.163",
#   "annual_burn_rate": "0.031",
#   "net_emission": "0.132",
#   "deflationary": false,
#   "deflationary_threshold_utilisation": "0.68",
#   "current_utilisation": "0.34",
#   "xqore_locked_fraction": "0.05",
#   "effective_velocity": "3.42",
#   "supply_trajectory": {
#     "year_5_projected_circ": "5234000000",
#     "year_10_projected_circ": "4987000000"
#   }
# }
```

11.7 Staking Economics

11.7.1 Combined Staking Returns

A QOR staker receives returns from three independent sources. The combined annual percentage yield (APY) for a delegator staking with validator v is:

$$\text{APY}_{\text{total}}(v) = \text{APY}_{\text{fee}}(v) + \text{APY}_{\text{bonding}}(v) + \text{APY}_{\text{emission}}(v) \tag{11.45}$$

Fee Distribution APY

The fee-based return for a staker with delegation d_i to validator v is:

$$\text{APY}_{\text{fee}}(v) = \frac{0.10 \cdot F_{\text{annual}}}{S_{\text{staked}}} + \frac{0.37 \cdot F_{\text{annual}} \cdot \frac{s_v}{\sum_j s_j} \cdot (1 - c_v)}{s_v} \tag{11.46}$$

where $c_v \in [0, 1]$ is the validator's commission rate. The first term is the passive staker share (10% of fees, distributed to all stakers), and the second term is the delegator's share of the validator's fee revenue (37% of fees, proportional to validator stake, minus commission).

Bonding Curve APY

The bonding curve reward (Equation 10.13) contributes:

$$\text{APY}_{\text{bonding}}(v) = \frac{R(v,t) \cdot N_{\text{epochs}} \cdot (1 - c_v)}{s_v} \tag{11.47}$$

This component rewards loyalty and reputation, meaning validators with longer delegation histories and higher reputation scores generate higher returns for their delegators.

Emission APY

The emission-based return depends on the inflation rate and staking ratio:

$$\text{APY}_{\text{emission}} = \frac{\epsilon(y) \cdot S_0 \cdot \alpha_{\text{staker}}}{S_{\text{staked}}} \tag{11.48}$$

where α_{staker} is the fraction of emissions directed to stakers. When the staking ratio is low ($S_{\text{staked}}/S_0 \ll 1$), the emission APY is high, incentivising more staking. As the staking ratio increases, the APY decreases, creating a natural equilibrium.

11.7.2 Staking Ratio Equilibrium

Let r_{stake} denote the required rate of return for a marginal staker (their opportunity cost). The equilibrium staking ratio ρ^* is the value at which $\text{APY}_{\text{total}} = r_{\text{stake}}$:

$$\rho^* = \frac{S^*_{\text{staked}}}{S_0} \quad \text{s.t.} \quad \text{APY}_{\text{total}}(\rho^*) = r_{\text{stake}} \tag{11.49}$$

Since APY decreases monotonically with ρ (more stakers dilute the rewards), the equilibrium exists and is unique for any $r_{\text{stake}} > 0$.

For the emission component alone:

$$\rho^*_{\text{emission}} = \frac{\epsilon(y) \cdot \alpha_{\text{staker}}}{r_{\text{stake}}} \tag{11.50}$$

In Year 1 with $\epsilon(1) = 0.175$ and $r_{\text{stake}} = 0.08$ (8% opportunity cost): $\rho^*_{\text{emission}} \approx 2.19 \cdot \alpha_{\text{staker}}$. With fee and bonding rewards added, the equilibrium staking ratio is higher, typically in the 40–60% range for mature Cosmos SDK chains.

11.7.3 Delegation Strategy Under QDRW

When the optional QDRW tally is activated, delegators who also hold xQORE receive additional governance influence. The optimal allocation between plain staking (s) and xQORE locking (x), subject to a total budget $B = s + x$, maximises the combined utility:

$$U(s,x) = \underbrace{\text{APY}_{\text{total}}(s) \cdot s}_{\text{staking return}} + \underbrace{\text{APY}_{\text{PvP}}(x) \cdot x}_{\text{xQORE return}} + \underbrace{\omega_{\text{gov}} \cdot VP(s,x,r)}_{\text{governance value}} \tag{11.51}$$

where ω_{gov} is the participant's subjective value for governance influence. Taking the first-order conditions:

$$\frac{\partial U}{\partial s} = \text{APY}_{\text{total}} + \frac{\omega_{\text{gov}} \cdot M(r)}{2\sqrt{s+2x} \quad \frac{\partial U}{\partial x} = \text{APY}_{\text{PvP}} + \frac{2\omega_{\text{gov}} \cdot M(r)}{2\sqrt{s+2x}}} \tag{11.52}$$

At the optimum:

$$\frac{x^*}{s^*} = \frac{2\omega_{\text{gov}} \cdot M(r) + \text{APY}_{\text{PvP}} \cdot 2\sqrt{B}}{\text{APY}_{\text{total}} \cdot 2\sqrt{B} + \omega_{\text{gov}} \cdot M(r)} \tag{11.53}$$

For participants who value governance highly ($\omega_{\text{gov}} \gg 0$), the optimal allocation favours xQORE (since it provides 2x governance weight per token). For pure yield seekers ($\omega_{\text{gov}} = 0$), the allocation depends solely on the relative APY of staking vs. xQORE PvP rebase.

11.7.4 APY Lifecycle Projection

Table 11.3: Projected Staking APY by Network Phase

Year	Emission APY	Fee APY	Bonding APY	Total APY
1 (Genesis)	12–18%	1–3%	2–4%	15–25%
2 (Growth)	7–11%	2–5%	2–3%	11–19%
3–4 (Stabilisation)	4–7%	3–6%	1–2%	8–15%
5+ (Mature)	1–2%	4–8%	1–2%	6–12%

The transition from emission-dominated returns (Year 1) to fee-dominated returns (Year 5+) reflects the design intent: early validators are compensated for bootstrapping the network, while long-term sustainability relies on organic transaction demand.

11.7.5 Risk-Adjusted Return Model

The Sharpe ratio of staking QOR, relative to a risk-free alternative, is:

$$\text{Sharpe}_{\text{QOR}} = \frac{\text{APY}_{\text{total}} - r_f}{\sigma_{\text{QOR}}} \tag{11.54}$$

where r_f is the risk-free rate and σ_{QOR} is the annualised volatility of QOR returns (including both staking yield and token price movement). For stakers who delegate to high-reputation validators with low slashing probability, the downside risk is primarily driven by token price volatility rather than protocol risk.

The expected loss from slashing, for a validator with infraction rate λ_{inf}, is:

$$\mathbb{E}[\text{SlashLoss}] = \lambda_{\text{inf}} \cdot \mathbb{E}[\text{Penalty}] \cdot s_v = \lambda_{\text{inf}} \cdot p_{\text{base}} \cdot \bar{\text{sev}} \cdot s_v \tag{11.55}$$

For a well-operated validator ($\lambda_{\text{inf}} < 0.001$ infractions/year, $p_{\text{base}} = 0.01$, $\bar{\text{sev}} = 1.0$): $\mathbb{E}[\text{SlashLoss}] < 0.001\%$ of stake annually, which is negligible relative to staking returns.

```
# Query estimated staking APY
qorechaind query staking apy-estimate --output json

# Example response:
# {
#   "current_year": 1,
#   "staking_ratio": "0.38",
#   "apy_components": {
#     "emission": "0.142",
#     "fee_distribution": "0.023",
#     "bonding_curve_avg": "0.031"
#   },
#   "total_apy_estimate": "0.196",
#   "projected_apy": {
#     "year_2": "0.145",
#     "year_3": "0.112",
#     "year_5": "0.085"
#   }
# }

# Query delegation rewards for a specific delegator
qorechaind query distribution rewards qor1abc...def --output json

# Query optimal staking/xQORE allocation
qorechaind query tokenomics optimal-allocation \
  --budget 1000000000000 \
  --governance-weight 0.5 \
  --output json
```

Chapter 12

On-Chain Governance

QoreChain implements a multi-tier on-chain governance system that enables token holders and validators to collectively manage protocol parameters, upgrade the network, and allocate community resources. At genesis, governance voting power is duration-weighted (staked QOR scaled by lock-duration multipliers of 1.0x to 2.0x, consistent with the QoreChain Tokenomics Paper); the Quadratic Delegated Reputation-Weighted (QDRW) framework introduced in Section **??** is available as a governance-activatable extension, disabled by default, that further distributes governance influence according to commitment and contribution rather than raw capital alone.

This chapter formalises the proposal lifecycle, voting mechanics, parameter safety constraints, emergency governance procedures, delegation dynamics, and treasury management. Each mechanism is accompanied by formal security analysis and game-theoretic characterisation.

12.1 Governance Architecture Overview

QoreChain governance operates on a three-tier structure, where each tier imposes different thresholds for quorum, approval, and voting duration. This tiered design reflects the principle that changes with greater systemic impact should require broader consensus and longer deliberation.

12.1.1 Tier Classification

Tier 1: Constitutional Proposals. These proposals affect foundational protocol parameters: consensus algorithm changes, PQC algorithm additions or deprecations, validator set size bounds, and token supply model modifications. Constitutional proposals require supermajority approval and extended deliberation.

Tier 2: Standard Proposals. These proposals cover routine governance actions: parameter adjustments within safety bounds, community pool spend requests, software upgrade signals, and text proposals. Standard proposals use simple majority with moderate quorum.

Tier 3: Operational (Expedited) Proposals. These proposals address time-sensitive operational matters: bridge circuit breaker overrides, emergency parameter corrections, and security incident responses. Operational proposals use elevated quorum but shortened voting periods to enable rapid response.

Each tier maps to distinct governance parameters (Section 12.4): quorum thresholds q_τ, approval thresholds θ_τ, voting duration d_τ in blocks, and deposit requirements D_τ in uqor. The tiering system recognizes that protocol stability depends on broad consensus for fundamental changes, moderate support for routine actions, and rapid governance for emergencies. Validators, delegators, and governance participants can instantiate proposals at any tier, subject to tier-specific deposit and voting power requirements. Upgrading a proposal across tiers (e.g., escalating a Standard proposal to Constitutional) is not permitted; the proposal tier is fixed at creation and cannot be changed.

12.1.2 Governance Parameters

The governance parameter set $\mathcal{G} = \{q_\tau, \theta_\tau, v_\tau, d_\tau, \eta_\tau\}_{\tau \in \{1,2,3\}}$ defines:

- q_τ: Quorum threshold (minimum fraction of total voting power that must participate)
- θ_τ: Approval threshold (minimum fraction of non-abstain votes that must be "Yes")
- v_τ: Veto threshold (maximum fraction of "NoWithVeto" votes before auto-rejection)
- d_τ: Voting period duration (in blocks)
- η_τ: Execution delay (blocks between passage and execution)

Table 12.1: Governance tier parameters

Tier	q_τ	θ_τ	v_τ	d_τ	η_τ
Constitutional (1)	0.50	0.667	0.334	100,800 (≈7 days)	28,800 (≈2 days)
Standard (2)	0.334	0.500	0.334	50,400 (≈3.5 days)	14,400 (≈1 day)
Operational (3)	0.45	0.667	0.334	14,400 (≈1 day)	0 (immediate)

12.1.3 Proposal Types

The governance module supports five proposal types, each mapped to a governance tier. Each type encodes specific constraints on voting duration, quorum thresholds, and approval supermajority to ensure that proposal complexity and impact align with decision-making rigor. Proposal types are immutable at protocol level, preventing governance from accidentally reducing safety thresholds for high-impact actions. Type mapping to tiers creates a natural escalation: signalling proposals require light coordination, parameter changes require moderate consensus, infrastructure decisions require strong consensus, and emergency actions require both speed and elevated participation. This design prevents a common governance failure mode wherein low-engagement voters make decisions about high-impact protocol changes without sufficient scrutiny.

1. **TextProposal** (Tier 2): Non-binding signalling proposals for community sentiment.

2. **ParameterChangeProposal** (Tier 1 or 2, depending on parameter category): Modify on-chain parameters within safety bounds.
3. **SoftwareUpgradeProposal** (Tier 1): Signal coordinated binary upgrades at a specified block height.
4. **CommunityPoolSpendProposal** (Tier 2): Allocate funds from the community treasury.
5. **EmergencyActionProposal** (Tier 3): Execute predefined emergency actions (bridge pause, module halt, circuit breaker override).

12.1.4 Formal Governance Model

Let $\mathcal{P}$ denote the set of all proposals. Each proposal $p \in \mathcal{P}$ is a tuple:

$$p = (\text{id}, \tau_p, \text{content}_p, \text{depositor}_p, t_{\text{submit}}, \text{state}_p) \tag{12.1}$$

where $\tau_p \in \{1, 2, 3\}$ is the governance tier. The proposal state machine $\mathcal{S}_p$ transitions through:

$$\mathcal{S}_p : \texttt{Deposit} \xrightarrow{\text{min deposit met}} \texttt{Voting} \xrightarrow{\text{tally}} \begin{cases} \texttt{Passed} \xrightarrow{\eta_\tau} \texttt{Executed} \\ \texttt{Rejected} \\ \texttt{Vetoed} \end{cases} \tag{12.2}$$

A proposal p of tier τ_p passes if and only if all three conditions hold simultaneously:

$$\text{Pass}(p) \iff \begin{cases} \displaystyle\sum_{i \in \mathcal{V}_p} VP_i \geq q_{\tau_p} \cdot VP_{\text{total}} & \text{(quorum)} \\ \dfrac{\sum_{i:v_i=\text{Yes}} VP_i}{\sum_{i:v_i \neq \text{Abstain}} VP_i} \geq \theta_{\tau_p} & \text{(approval)} \\ \dfrac{\sum_{i:v_i=\text{NWV}} VP_i}{\sum_{i \in \mathcal{V}_p} VP_i} < v_{\tau_p} & \text{(no veto)} \end{cases} \tag{12.3}$$

where $\mathcal{V}_p$ is the set of voters on proposal p, VP_i is the QDRW voting power of voter i, and $v_i \in \{\text{Yes}, \text{No}, \text{NoWithVeto}, \text{Abstain}\}$ is the vote cast.

```
# Submit a parameter change proposal (Tier 2)
qorechaind tx gov submit-proposal param-change \
  proposal.json \
  --from mykey \
  --deposit 10000000000uqor \
  --output json

# Query active proposals
qorechaind query gov proposals --status voting_period --output json

# Vote on a proposal
qorechaind tx gov vote 42 yes --from mykey --output json

# Query proposal tally in real time
qorechaind query gov tally 42 --output json
```

12.2 QDRW Voting Power in Governance

The Quadratic Delegated Reputation-Weighted (QDRW) model, introduced in Section **??**, is an optional tally extension that is disabled at genesis; until activated by governance proposal, voting uses the duration-weighted model (lock multipliers 1.0x to 2.0x). This section provides the complete governance-focused analysis of the QDRW mechanism as it applies once activated.

12.2.1 Voting Power Function

Each governance participant i with staked balance s_i, xQORE balance x_i, and reputation score $r_i \in [0, 1]$ receives voting power:

$$VP_i = \sqrt{s_i + 2x_i} \cdot Q(r_i) \tag{12.4}$$

where $Q(r_i)$ is the reputation multiplier:

$$Q(r_i) = Q_{\min} + (Q_{\max} - Q_{\min}) \cdot r_i = 0.5 + 1.5r_i \tag{12.5}$$

mapping reputation to the range $[Q_{\min}, Q_{\max}] = [0.5, 2.0]$.

12.2.2 Whale Dampening Analysis

The square root function provides sub-linear scaling of voting power with respect to capital. For two participants with capital ratio $k = c_2/c_1$:

$$\frac{VP_2}{VP_1} = \sqrt{k} \cdot \frac{Q(r_2)}{Q(r_1)} \tag{12.6}$$

A participant with 100x the capital of another obtains only $\sqrt{100} = 10$x the base voting power. With equal reputation, this represents a 90% dampening of the capital advantage. If the larger holder has poor reputation ($r = 0$, giving $Q = 0.5$) and the smaller holder has excellent reputation ($r = 1$, giving $Q = 2.0$):

$$\frac{VP_{\text{whale}}}{VP_{\text{small}}} = \sqrt{100} \cdot \frac{0.5}{2.0} = 10 \cdot 0.25 = 2.5 \tag{12.7}$$

A 100x capital advantage is reduced to a 2.5x governance influence ratio, representing a 97.5% dampening effect.

12.2.3 xQORE Governance Efficiency

The marginal voting power per unit of capital allocated to xQORE versus plain staking:

$$\frac{\partial VP}{\partial x} = \frac{Q(r)}{\sqrt{s + 2x}, \qquad \frac{\partial VP}{\partial s} = \frac{Q(r)}{2\sqrt{s+2x}}} \tag{12.8}$$

The ratio $\frac{\partial VP/\partial x}{\partial VP/\partial s} = 2$ confirms that each unit of xQORE contributes exactly twice the marginal voting power of plain staking. This differential creates a rational incentive to lock tokens into xQORE for governance participation, which simultaneously reduces circulating supply and signals long-term commitment.

12.2.4 Lorenz Curve and Gini Coefficient

To quantify governance power distribution, define the Lorenz curve $L(\cdot)$ over the sorted voting power distribution. For n participants with QDRW voting powers $VP_{(1)} \leq VP_{(2)} \leq \cdots \leq VP_{(n)}$:

$$L\left(\frac{k}{n}\right) = \frac{\sum_{i=1}^{k} VP_{(i)}}{\sum_{i=1}^{n} VP_{(i)}}, \qquad k = 0, 1, \ldots, n \tag{12.9}$$

The Gini coefficient is:

$$G = 1 - 2\int_0^1 L(p)\,dp = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} |VP_i - VP_j|}{2n\sum_{i=1}^{n} VP_i} \tag{12.10}$$

Under linear voting ($VP_i = s_i$), the Gini coefficient equals the Gini of the stake distribution $G_{\text{linear}} = G_s$. Under QDRW:

$$G_{\text{QDRW}} < G_{\text{linear}} \tag{12.11}$$

This inequality holds because the square root is a concave function, and applying a concave transformation to a non-negative distribution strictly reduces the Gini coefficient (by the Schur-concavity of the Gini with respect to Lorenz-dominating distributions). The reputation multiplier further reduces concentration when reputation is not perfectly correlated with stake.

12.2.5 Nakamoto Coefficient

The Nakamoto coefficient N_c is the minimum number of participants whose combined voting power exceeds one-third of the total:

$$N_c = \min\left\{k : \sum_{i=1}^{k} VP_{(n-i+1)} > \frac{1}{3}\sum_{j=1}^{n} VP_j\right\} \tag{12.12}$$

Higher N_c indicates greater decentralisation. Since QDRW compresses the top of the voting power distribution relative to linear voting:

$$N_c^{\text{QDRW}} \geq N_c^{\text{linear}} \tag{12.13}$$

with strict inequality whenever the top-k holders have stake shares exceeding their square-root-weighted shares.

```
# Query governance voting power for an account
qorechaind query gov voting-power qor1abc...def --output json

# Example response:
# {
#   "address": "qor1abc...def",
#   "staked": "5000000000000",
#   "xqore_balance": "2000000000000",
#   "reputation_score": "0.85",
#   "reputation_multiplier": "1.775",
#   "base_voting_power": "3000.0",
#   "weighted_voting_power": "5325.0",
```

```
#   "pct_of_total": "0.0142"
# }

# Query governance distribution metrics
qorechaind query gov distribution-metrics --output json

# Example response:
# {
#   "gini_coefficient": "0.412",
#   "gini_linear_comparison": "0.687",
#   "nakamoto_coefficient": 14,
#   "nakamoto_linear_comparison": 5,
#   "total_voting_power": "375000.0",
#   "active_voters": 1847
# }
```

12.3 Proposal Lifecycle and Mechanisms

12.3.1 Deposit Phase

Every proposal begins in the `Deposit` state. The proposer must provide an initial deposit $d_0 \geq d_{\min}^{\text{init}}$ (minimum initial deposit). Other participants may contribute additional deposits during the deposit window of W_d blocks. The proposal advances to `Voting` if and only if the total deposit reaches the minimum deposit threshold $D_{\min}$ before the window expires:

$$\sum_j d_j \geq D_{\min}(\tau_p) \quad \text{within } W_d \text{ blocks of submission} \tag{12.14}$$

Deposit thresholds scale with governance tier:

$$D_{\min}(\tau) = D_{\text{base}} \cdot \mu_\tau, \qquad \mu_1 = 5, \quad \mu_2 = 1, \quad \mu_3 = 3 \tag{12.15}$$

where D_{base} is the base deposit (governance-configurable, default: 10,000 QOR). Constitutional proposals require 5x the base deposit to deter frivolous fundamental changes; operational proposals require 3x to prevent spam of emergency actions while remaining accessible during genuine emergencies.

Deposit Refund Rule. If the proposal reaches the voting phase and is not vetoed, all deposits are refunded. If the proposal fails to reach quorum or is vetoed (NoWithVeto exceeds v_τ), deposits are burned. This creates a game-theoretic filter:

$$\mathbb{E}[\text{Deposit Return}] = P(\text{quorum} \cap \neg\text{veto}) \cdot D - P(\neg\text{quorum} \cup \text{veto}) \cdot D \tag{12.16}$$

A rational depositor contributes only when $P(\text{quorum} \cap \neg\text{veto}) > 0.5$, filtering out proposals likely to lack community support.

12.3.2 Deposit Contribution Game

The deposit phase can be modelled as a public goods game. Let N potential depositors each decide whether to contribute d_j toward the threshold $D_{\min}$. Define the payoff for depositor j:

$$u_j(d_j, d_{-j}) = \begin{cases} B_j - d_j + d_j \cdot \mathbf{1}[\text{not vetoed}] & \text{if } \sum_k d_k \geq D_{\min} \\ -d_j & \text{if } \sum_k d_k < D_{\min} \text{ (deposit burned on timeout)} \end{cases} \tag{12.17}$$

where B_j is depositor j's private benefit from the proposal passing. The threshold public goods game has a Nash equilibrium where the proposal reaches `Voting` if and only if $\sum_j B_j > D_{\min}$, provided depositors can coordinate (which is facilitated by the on-chain deposit visibility).

12.3.3 Voting Phase

During the voting window of d_τ blocks, each eligible voter casts exactly one vote $v_i \in \{\text{Yes}, \text{No}, \text{NoWithVeto}, \text{Abstain}\}$, weighted by their governance voting power VP_i (duration-weighted at genesis; QDRW if activated). Voters may change their vote at any time before the window closes. The final tally uses the vote recorded at the closing block.

Vote Options.

Yes: Support the proposal. *No*: Oppose the proposal. *NoWithVeto*: Strong opposition indicating the proposal is harmful or spam; contributes to the veto threshold and, if sufficient, burns the deposit. *Abstain*: Participate for quorum purposes without expressing preference; abstain votes count toward quorum but not toward the approval ratio.

Vote changes are tracked on-chain for transparency; the governance module emits events whenever a vote is cast or modified. This allows governance interfaces to show real-time tally dynamics. Voters with zero voting power (having unstaked all xQOR and liquidated governance NFTs) are ineligible to vote; attempts to vote return an error without consuming gas.

The voting window duration d_τ is parameterised per tier: Tier 1 typically uses 14 days (approximately 1,209,600 blocks at 6-second block time), Tier 2 uses 7 days, and Tier 3 uses 2 days. These durations can be adjusted through Tier 1 governance. Vote changes incur minimal on-chain cost (10 uqor) to prevent vote-spam attacks.

12.3.4 Tally Function

At the close of the voting period, the tally function $\mathcal{T}(p)$ computes:

$$\text{TotalVP} = \sum_{i \in \mathcal{V}_p} VP_i \tag{12.18}$$

$$\text{YesVP} = \sum_{i : v_i = \text{Yes}} VP_i \tag{12.19}$$

$$\text{NoVP} = \sum_{i : v_i = \text{No}} VP_i \tag{12.20}$$

$$\text{NwvVP} = \sum_{i : v_i = \text{NWV}} VP_i \tag{12.21}$$

$$\text{AbstainVP} = \sum_{i : v_i = \text{Abstain}} VP_i \tag{12.22}$$

The outcome is determined by the following decision procedure:

$$\mathcal{T}(p) = \begin{cases} \texttt{QuorumFailed} & \text{if TotalVP} < q_\tau \cdot VP_{\text{total}} \\ \texttt{Vetoed} & \text{if NwvVP/TotalVP} \geq v_\tau \\ \texttt{Passed} & \text{if YesVP/(TotalVP} - \text{AbstainVP)} \geq \theta_\tau \\ \texttt{Rejected} & \text{otherwise} \end{cases} \tag{12.23}$$

The evaluation order matters: quorum is checked first, then veto, then approval. A proposal that meets quorum but is vetoed does not pass regardless of Yes votes.

12.3.5 Execution Delay and Finality

Passed proposals enter an execution delay of η_τ blocks before their content is applied on-chain. This delay serves two purposes: it allows validators and users to prepare for parameter changes or upgrades, and it provides a window for emergency governance to cancel a proposal if a critical issue is discovered post-passage.

For Tier 3 (operational) proposals, $\eta_3 = 0$, meaning execution is immediate upon passage. This is necessary for security incident response but requires the elevated quorum ($q_3 = 0.45$) to compensate for the reduced deliberation time.

```
# Submit a community pool spend proposal
qorechaind tx gov submit-proposal community-pool-spend \
  --title "Developer Grant: Cross-Chain SDK" \
  --description "Fund development of cross-chain SDK tooling" \
  --recipient qor1grant...addr \
  --amount 500000000000uqor \
  --deposit 10000000000uqor \
  --from mykey \
  --output json

# Query proposal status and deposit progress
qorechaind query gov proposal 42 --output json

# Example response:
# {
#   "proposal_id": "42",
```

```
#   "tier": 2,
#   "status": "PROPOSAL_STATUS_DEPOSIT_PERIOD",
#   "total_deposit": "7500000000uqor",
#   "min_deposit": "10000000000uqor",
#   "deposit_end_time": "2026-04-15T12:00:00Z",
#   "content": {
#     "@type": "/cosmos.distribution.v1beta1.CommunityPoolSpendProposal",
#     "title": "Developer Grant: Cross-Chain SDK",
#     "amount": "500000000000uqor"
#   }
# }
```

12.4 Parameter Governance and Safety Bounds

On-chain parameter changes are one of the most powerful governance actions, as they directly alter protocol behaviour. QoreChain constrains parameter governance through a formal safety bound system that prevents governance from inadvertently destabilising the network.

12.4.1 Parameter Categories

All governance-modifiable parameters are classified into three categories based on their systemic impact:

Category A: Consensus Parameters (Tier 1 governance). Block size, maximum gas per block, validator set size, epoch length, RL agent mode. These parameters directly affect consensus safety and liveness.

Category B: Economic Parameters (Tier 2 governance). Emission decay rates, fee distribution percentages, burn channel weights, slashing base penalty, bonding curve coefficients. These parameters affect economic incentives but not consensus safety directly.

Category C: Operational Parameters (Tier 2 governance). Bridge confirmation depths, circuit breaker thresholds, minimum deposit amounts, voting period durations. These parameters affect user experience and operational safety.

The categorisation principle reflects the impact radius: Category A changes alter protocol invariants and require the highest governance bar. Category B changes affect token economics and validator incentives but preserve safety properties. Category C changes tune operational thresholds without affecting core mechanics. Each category has a distinct governance workflow, deposit requirement, and voting threshold.

Parameter changes within Category B and C are subject to maximum change-rate bounds $\Delta_k^{\max}$ enforced by the parameter safety module. For example, the burn percentage can change by at most 5 percentage points per proposal, preventing governance from accidentally reducing burn to 0%. Similarly, the validator set size can grow by at most 50 validators per Tier 1 proposal, preventing rapid expansion that could degrade finality latency. These bounds are configurable through Tier 1 governance but provide strong default protections.

12.4.2 Safety Bound Formalism

Each parameter ρ_k has a governance-enforced safety interval $[\rho_k^{\min}, \rho_k^{\max}]$ and a maximum per-proposal change rate $\Delta_k^{\max}$:

$$\rho_k^{\min} \leq \rho_k^{\text{new}} \leq \rho_k^{\max} \quad \wedge \quad |\rho_k^{\text{new}} - \rho_k^{\text{current}}| \leq \Delta_k^{\max} \tag{12.24}$$

Any `ParameterChangeProposal` that violates these bounds is rejected at the message validation layer, before entering the deposit phase.

12.4.3 Parameter Interdependency Constraints

Some parameters are interdependent. The fee distribution parameters $\{f_v, f_b, f_t, f_s, f_l\}$ (validator, burn, treasury, staker, light node shares) must satisfy:

$$f_v + f_b + f_t + f_s + f_l = 1.0, \qquad f_k \geq 0 \ \ \forall k \tag{12.25}$$

This simplex constraint is enforced at the proposal validation layer. A parameter change proposal that modifies any fee share must specify all five values, and the proposal is rejected if the sum deviates from 1.0.

12.4.4 Lyapunov Stability of Parameter Trajectories

To ensure governance-driven parameter changes do not cause oscillatory or divergent system behaviour, consider the system state $\mathbf{x}(t) = [\text{TPS}(t), \text{StakeRatio}(t), \text{FeeRate}(t), \ldots]^\top$ as a function of parameter vector $\boldsymbol{\rho}(t)$. Define a Lyapunov candidate function:

$$V(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}^*)^\top \mathbf{P}(\mathbf{x} - \mathbf{x}^*) \tag{12.26}$$

where $\mathbf{x}^*$ is the target equilibrium and $\mathbf{P} \succ 0$ is a positive definite weighting matrix. The maximum change rate constraint $\Delta_k^{\max}$ ensures:

$$\dot{V}(\mathbf{x}) = (\mathbf{x} - \mathbf{x}^*)^\top \mathbf{P} \frac{\partial \mathbf{x}}{\partial \boldsymbol{\rho}} \dot{\boldsymbol{\rho}} \leq 0 \tag{12.27}$$

provided $\|\dot{\boldsymbol{\rho}}\|_\infty \leq \max_k \Delta_k^{\max} / T_{\text{vote}}$ is sufficiently small relative to the system's natural response time. The per-proposal change rate limits thus serve as a discrete-time analogue of a Lyapunov stability guarantee: the system can absorb the perturbation from each governance action before the next takes effect.

12.4.5 Constraint Satisfaction

The feasible parameter space $\mathcal{F}$ is the intersection of all safety intervals and interdependency constraints:

$$\mathcal{F} = \left\{ \boldsymbol{\rho} \in \mathbb{R}^m : \rho_k^{\min} \leq \rho_k \leq \rho_k^{\max} \ \ \forall k, \ \ \mathbf{A}\boldsymbol{\rho} = \mathbf{b}, \ \ \mathbf{C}\boldsymbol{\rho} \leq \mathbf{d} \right\} \tag{12.28}$$

where $\mathbf{A}\boldsymbol{\rho} = \mathbf{b}$ captures equality constraints (such as fee distribution summing to 1.0) and $\mathbf{C}\boldsymbol{\rho} \leq \mathbf{d}$ captures inequality constraints (such as minimum staking APY floors). Every parameter change proposal must map the current parameter vector to a new vector within $\mathcal{F}$, and the transition must satisfy the per-step change rate bounds.

```
# Query safety bounds for a parameter
qorechaind query params safety-bounds staking MaxValidators --output json

# Example response:
# {
#   "parameter": "staking.MaxValidators",
#   "category": "A",
#   "governance_tier": 1,
#   "current_value": "150",
#   "min_bound": "50",
#   "max_bound": "500",
#   "max_change_per_proposal": "25",
#   "last_changed_proposal": 31,
#   "last_changed_block": 4521000
# }

# Validate a parameter change proposal before submission
qorechaind tx gov check-proposal param-change proposal.json --output json
```

12.5 Emergency Governance

Security incidents and critical protocol failures require governance responses faster than standard proposal timelines allow. QoreChain provides two complementary emergency mechanisms: expedited Tier 3 proposals and a guardian multisig with mandatory post-hoc ratification.

12.5.1 Expedited Proposals (Tier 3)

Tier 3 proposals use a 1-day voting period ($d_3 = 14{,}400$ blocks) with elevated quorum ($q_3 = 0.45$) and supermajority approval ($\theta_3 = 0.667$). The compressed timeline and elevated participation requirement reflect the high-stakes nature of emergency decisions: while faster governance is needed to respond to active threats, accelerated decision-making must be counterbalanced by higher consensus thresholds to prevent hasty governance mistakes. The action whitelist is immutable by governance vote, preventing emergency governance from being weaponized to expand its own powers. This design prevents a scenario wherein a network crisis could be used to justify seizing governance power that would normally require longer deliberation. The whitelist is updated only through standard Tier 1 proposals, maintaining the principle that extraordinary powers require extraordinary consensus. The available emergency actions are restricted to a predefined set:

1. **BridgePause**: Halt outbound transfers on a specified bridge endpoint.
2. **ModuleHalt**: Pause transaction processing for a specified module.
3. **CircuitBreakerOverride**: Manually trigger or reset a circuit breaker.
4. **ValidatorJail**: Immediately jail a validator pending investigation.

5. **ParameterRevert**: Revert a parameter to its previous value.

Tier 3 proposals cannot introduce new parameters, upgrade binaries, or spend treasury funds. This restriction limits the attack surface of expedited governance.

12.5.2 Guardian Multisig

For incidents requiring sub-block response times (active bridge exploits, consensus halts), a guardian multisig provides immediate action capability. The guardian set $\mathcal{G}_{\text{guard}}$ consists of n_g members with a threshold k_g-of-n_g signature requirement:

$$k_g = \left\lceil \frac{2}{3} n_g \right\rceil + 1 \tag{12.29}$$

Guardian actions are limited to defensive operations: bridge pause, module halt, and emergency parameter revert. Critically, every guardian action automatically generates a Tier 3 ratification proposal that must pass within 7 days, or the guardian action is automatically reverted:

$$\text{GuardianAction}(a) \xrightarrow{\text{auto}} \text{Tier3Proposal}(\text{Ratify}(a),\ d_{\text{ratify}} = 100{,}800) \tag{12.30}$$

This mechanism ensures that the guardian multisig cannot permanently alter protocol state without community consent.

12.5.3 Expected Loss Analysis

The value of emergency governance can be quantified by comparing expected losses under standard versus expedited response. Let $L(t)$ be the cumulative loss function for an active exploit, and T_{std}, T_{exp}, T_{guard} be the response times for standard, expedited, and guardian governance respectively:

$$\Delta L = L(T_{\text{std}}) - L(T_{\text{guard}}) = \int_{T_{\text{guard}}}^{T_{\text{std}}} \ell(t)\, dt \tag{12.31}$$

where $\ell(t)$ is the instantaneous loss rate. For a bridge exploit draining funds at rate ℓ:

$$\Delta L = \ell \cdot (T_{\text{std}} - T_{\text{guard}}) \approx \ell \cdot (3.5 \text{ days} - 10 \text{ minutes}) \approx 3.49 \cdot \ell \text{ days} \tag{12.32}$$

This quantifies the economic justification for maintaining the guardian multisig: the prevented loss must exceed the centralisation risk introduced by the multisig, which is mitigated by the mandatory ratification requirement.

12.5.4 Guardian Decentralisation Path

The guardian multisig is designed as a transitional mechanism. As the validator set matures and governance participation rates increase, the guardian threshold can be raised and eventually deprecated through a Constitutional (Tier 1) proposal. The planned progression:

$$\text{Phase 1} : k_g = \lceil 2n_g/3 \rceil + 1, \quad \text{Phase 2} : k_g = n_g - 1, \quad \text{Phase 3} : \text{Guardian deprecated} \tag{12.33}$$

```
# Submit an emergency bridge pause proposal
qorechaind tx gov submit-proposal emergency-action \
  --action bridge-pause \
  --target-chain ethereum \
  --reason "Anomalous outflow detected" \
  --deposit 30000000000uqor \
  --from mykey \
  --output json

# Query guardian multisig status
qorechaind query gov guardian-status --output json

# Example response:
# {
#   "guardian_set_size": 7,
#   "threshold": 5,
#   "active_actions": [
#     {
#       "action_id": 3,
#       "type": "bridge_pause",
#       "target": "ethereum",
#       "executed_block": 5120300,
#       "ratification_proposal_id": 88,
#       "ratification_deadline_block": 5221100,
#       "status": "awaiting_ratification"
#     }
#   ]
# }
```

12.6 Delegation and Liquid Governance

Staking delegation in QoreChain extends beyond economic participation into governance representation. Validators act as default governance delegates for their delegators, but the system supports granular vote override and delegation of governance power independently of staking.

12.6.1 Default Delegation

When a delegator j stakes to validator i and does not cast a direct vote on proposal p, the validator's vote is inherited with weight proportional to the delegator's contribution to the validator's total voting power. If the delegator votes directly, their vote overrides the validator's inherited vote for their share.

Let $VP_j^{(i)}$ denote the voting power delegated from j to validator i. The effective vote tally for option o is:

$$\mathrm{VP}_o = \sum_{i:v_i=o} \left(VP_i^{\text{self}} + \sum_{j \in \mathcal{D}_i^{\text{inherit}}} VP_j^{(i)} \right) + \sum_{j:v_j^{\text{direct}}=o} VP_j \tag{12.34}$$

where $\mathcal{D}_i^{\text{inherit}} = \{j \in \mathcal{D}_i : j \notin \mathcal{V}_p^{\text{direct}}\}$ is the set of delegators to i who did not vote directly, and $\mathcal{V}_p^{\text{direct}}$ is the set of all direct voters.

12.6.2 Split Voting

QoreChain supports weighted split voting, allowing a participant to distribute their voting power across multiple options:

$$v_i = \{(o_1, w_1), (o_2, w_2), \ldots\}, \qquad \sum_k w_k = 1, \quad w_k \geq 0 \tag{12.35}$$

For example, a participant may vote 70% Yes and 30% Abstain, reflecting partial confidence. Split voting is particularly useful for validators representing diverse delegator interests:

$$VP_{i,o} = w_{i,o} \cdot VP_i \tag{12.36}$$

12.6.3 Principal-Agent Model of Delegation

Governance delegation introduces a principal-agent dynamic. The delegator (principal) delegates governance authority to the validator (agent), creating potential preference misalignment. Let θ_j be the delegator's ideal policy position and θ_i be the validator's position. The delegator's utility under delegation:

$$U_j^{\text{delegate}} = -(\theta_j - \theta_i)^2 - c_{\text{monitor}} \tag{12.37}$$

where c_{monitor} is the cost of monitoring the validator's governance votes. The utility under direct voting:

$$U_j^{\text{direct}} = 0 - c_{\text{vote}} \tag{12.38}$$

where c_{vote} is the cost of evaluating and casting a vote on each proposal. A delegator rationally votes directly when:

$$(\theta_j - \theta_i)^2 > c_{\text{vote}} - c_{\text{monitor}} \tag{12.39}$$

This implies that delegation is efficient when the delegator's preferences are closely aligned with the validator's (small $|\theta_j - \theta_i|$) and when voting costs are high relative to monitoring costs. QoreChain's transparent on-chain voting history reduces c_{monitor}, lowering the barrier to direct participation.

12.6.4 Optimal Delegation Strategy

A delegator distributing stake across K validators $\{i_1, \ldots, i_K\}$ with governance alignment scores $a_k = 1 - (\theta_j - \theta_{i_k})^2 / \max_k (\theta_j - \theta_{i_k})^2$ and staking returns r_k solves:

$$\max_{\mathbf{w}} \ \alpha_{\text{econ}} \sum_k w_k r_k + \alpha_{\text{gov}} \sum_k w_k a_k \qquad \text{s.t.} \quad \sum_k w_k = 1, \quad w_k \geq 0 \tag{12.40}$$

where α_{econ} and α_{gov} are the delegator's weights on economic versus governance objectives. This is a linear program with solution at a vertex of the simplex, meaning the optimal strategy concentrates delegation on the validator(s) offering the best combined economic-governance value. In practice, diversification across 2–3 validators is common due to slashing risk considerations not captured in this simplified model.

```
# Cast a split vote on a proposal
qorechaind tx gov weighted-vote 42 \
  "yes=0.7,abstain=0.3" \
  --from myvalidator \
  --output json

# Query governance voting history for a validator
qorechaind query gov votes-by-voter qorvaloper1abc...def --output json

# Query delegation governance alignment
qorechaind query gov delegation-alignment \
  --delegator qor1abc...def \
  --output json

# Example response:
# {
#   "delegator": "qor1abc...def",
#   "delegations": [
#     {
#       "validator": "qorvaloper1xyz...ghi",
#       "stake_share": "0.60",
#       "governance_alignment": "0.92",
#       "proposals_voted": 38,
#       "override_rate": "0.05"
#     },
#     {
#       "validator": "qorvaloper1mno...pqr",
#       "stake_share": "0.40",
#       "governance_alignment": "0.78",
#       "proposals_voted": 35,
#       "override_rate": "0.17"
#     }
#   ]
# }
```

12.7 Treasury and Community Pool

The QoreChain treasury is funded by a continuous 20% allocation of all transaction fees (Section 11.4). The treasury serves as the protocol's self-sustaining funding mech-

anism for development grants, security audits, ecosystem incentives, and operational expenses.

12.7.1 Treasury Inflow Model

Let $F(t)$ denote the total fees collected at block t. The treasury inflow at block t is:

$$I_T(t) = f_t \cdot F(t) = 0.20 \cdot F(t) \tag{12.41}$$

The cumulative treasury balance at block T (assuming no outflows) is:

$$B_T(T) = B_T(0) + \sum_{t=1}^{T} I_T(t) = B_T(0) + 0.20 \sum_{t=1}^{T} F(t) \tag{12.42}$$

Under steady-state fee generation with average fee per block $\bar{F}$, the annualised treasury inflow is:

$$I_T^{\text{annual}} = 0.20 \cdot \bar{F} \cdot B_{\text{year}} \tag{12.43}$$

where $B_{\text{year}} \approx 5{,}256{,}000$ is the approximate number of blocks per year (at 6-second block time).

12.7.2 Community Pool Spend Mechanism

Treasury funds are disbursed through `CommunityPoolSpendProposal` (Tier 2 governance). Each proposal specifies a recipient address and amount. The governance tally determines whether the spend is approved.

The treasury acts as the protocol's development fund, controlled collectively by QOR token holders. Spend proposals are evaluated against the treasury's current balance and long-term sustainability. The treasury receives 20% of all transaction fees (Equation 11.19) and accumulates across epochs. Under steady-state operation with average fee generation $\bar{F}$ per block, the annualised inflow is $0.20 \cdot \bar{F} \cdot 5{,}256{,}000$ uqor per year.

Spend mechanics require governance participants to estimate the ecosystem value created by a grant. Typical grant categories include: (1) core development and protocol improvements, (2) tooling and infrastructure for developers, (3) research on cryptography and consensus, (4) community and marketing initiatives, and (5) emergency response and bug bounties. Each category has a soft spending cap (governance-recommended but not enforced), guiding allocations without hard constraints.

The spend approval process uses a public goods game framework. Once a spend proposal is approved, the treasury module transfers the requested amount to the recipient address via a `MsgSend` message. If a proposal is rejected or withdrawn before voting closes, the proposal deposit is refunded. Spending is irreversible once executed; there is no built-in mechanism to recover treasury funds from a recipient address, placing responsibility on governance participants to evaluate proposals carefully before voting.

12.7.3 Optimal Grant Sizing

The treasury faces a classic resource allocation problem: how to size grants to maximise ecosystem value subject to budget sustainability. Let $\{g_1, \ldots, g_M\}$ be a set of

grant proposals with estimated ecosystem values $\{V_1, \ldots, V_M\}$ and requested amounts $\{a_1, \ldots, a_M\}$. The treasury allocation problem is:

$$\max_{\mathbf{z} \in \{0,1\}^M} \sum_{m=1}^{M} z_m V_m \qquad \text{s.t.} \quad \sum_{m=1}^{M} z_m a_m \leq B_T^{\text{available}} \cdot \kappa \tag{12.44}$$

where $\kappa \in (0, 1)$ is the maximum spend-to-balance ratio per governance cycle (preventing treasury depletion in a single cycle). This is a 0-1 knapsack problem, solvable by governance through sequential proposal evaluation, where each approved proposal reduces the remaining budget for subsequent proposals in the same cycle.

12.7.4 Treasury Sustainability Condition

The treasury is sustainable if long-term inflows exceed long-term outflows:

$$\mathbb{E}[I_T^{\text{annual}}] \geq \mathbb{E}[O_T^{\text{annual}}] \tag{12.45}$$

Substituting the inflow model and assuming outflows grow with ecosystem size:

$$0.20 \cdot \mathbb{E}[\bar{F}] \cdot B_{\text{year}} \geq O_{\text{base}} + \omega \cdot \text{EcosystemSize}(t) \tag{12.46}$$

where O_{base} is fixed operational cost and ω is the marginal cost of ecosystem growth. As network usage grows, fee revenue scales with transaction volume, providing a natural sustainability mechanism: the treasury's income grows proportionally with the ecosystem it funds.

```
# Query treasury balance and inflow statistics
qorechaind query distribution community-pool --output json

# Example response:
# {
#   "pool": [
#     {
#       "denom": "uqor",
#       "amount": "2847563000000.000000000000000000"
#     }
#   ],
#   "inflow_30d": "142500000000",
#   "outflow_30d": "50000000000",
#   "pending_proposals": 2,
#   "pending_amount": "75000000000"
# }

# Query treasury sustainability metrics
qorechaind query distribution treasury-health --output json

# Example response:
# {
#   "balance_uqor": "2847563000000",
#   "avg_daily_inflow": "4750000000",
#   "avg_daily_outflow": "1666666666",
#   "sustainability_ratio": "2.85",
```

```
#   "runway_days_at_current_rate": 1281,
#   "spend_to_balance_ratio": "0.018"
# }
```

12.8 Formal Security Analysis

12.8.1 Bribery Attack Cost

A bribery attack on governance requires the attacker to purchase sufficient voting power to pass a malicious proposal. Under the optional QDRW tally (when activated), the cost of acquiring voting power VP_{target} through token purchase is:

$$C_{\text{bribe}}^{\text{QDRW}} = \left(\frac{VP_{\text{target}}}{Q(r_{\text{attacker}})} \right)^2 \cdot p_{\text{QOR}} \tag{12.47}$$

where p_{QOR} is the per-token price. Since an attacker has no established reputation ($r_{\text{attacker}} \approx 0$, giving $Q = 0.5$), the cost to achieve a given voting power level is:

$$C_{\text{bribe}}^{\text{QDRW}} = \left(\frac{VP_{\text{target}}}{0.5} \right)^2 \cdot p_{\text{QOR}} = 4 \cdot VP_{\text{target}}^2 \cdot p_{\text{QOR}} \tag{12.48}$$

Compare with linear voting where $C_{\text{bribe}}^{\text{linear}} = VP_{\text{target}} \cdot p_{\text{QOR}}$. The QDRW bribery cost scales quadratically with the target voting power, making large-scale bribery exponentially more expensive:

$$\frac{C_{\text{bribe}}^{\text{QDRW}}}{C_{\text{bribe}}^{\text{linear}}} = \frac{4 \cdot VP_{\text{target}}^2}{VP_{\text{target}}} = 4 \cdot VP_{\text{target}} \tag{12.49}$$

For a governance attack requiring $VP_{\text{target}} = 1{,}000{,}000$ units, QDRW makes the attack $4{,}000{,}000\times$ more expensive than under linear voting.

12.8.2 Flash-Loan Governance Prevention

Flash-loan governance attacks, where an attacker borrows tokens to vote and returns them within the same block, are prevented by two mechanisms:

xQORE Lock Requirement. Since xQORE provides 2x governance efficiency and requires a minimum 90-day lock (with graduated exit penalties), meaningful governance influence requires long-term capital commitment that is incompatible with flash loans.

Staking Unbonding Period. Staked tokens (the other source of voting power) have a 21-day unbonding period. Tokens must be staked before the proposal enters the voting phase to count toward the tally.

Together, these mechanisms ensure that governance voting power represents genuine long-term economic commitment:

$$\text{MinLockDuration}(VP) = \begin{cases} 21 \text{ days} & \text{if } VP \text{ from staking only} \\ 90 \text{ days} & \text{if } VP \text{ includes xQORE} \end{cases} \tag{12.50}$$

12.8.3 Governance Capture Probability

Define governance capture as a single entity controlling more than $\theta_1 = 2/3$ of voting power. Under QDRW with n active participants, the probability of capture by an entity with fraction ϕ of total token supply:

$$P(\text{capture}) = P\left(\frac{\sqrt{\phi \cdot S_{\text{total}}} + 2x_{\text{attacker}} \cdot Q(r_{\text{attacker}})}{\sum_{i=1}^{n} VP_i} \geq \theta_1\right) \tag{12.51}$$

For an attacker with no reputation ($Q = 0.5$) and no xQORE ($x = 0$):

$$P(\text{capture}) = P\left(\frac{0.5\sqrt{\phi \cdot S_{\text{total}}}}{\text{VP}_{\text{total}}} \geq \frac{2}{3}\right) \tag{12.52}$$

Solving for the minimum supply fraction needed:

$$\phi_{\min} = \frac{(4/3 \cdot \text{VP}_{\text{total}})^2}{S_{\text{total}}} \tag{12.53}$$

With active governance participation, VP_{total} is substantial, making $\phi_{\min}$ prohibitively large. If $\text{VP}_{\text{total}} = 50{,}000$ and $S_{\text{total}} = 4.5 \times 10^9$:

$$\phi_{\min} = \frac{(66{,}667)^2}{4.5 \times 10^9} \approx 0.000988 \approx 0.1\% \tag{12.54}$$

However, this assumes the attacker's tokens are the only staked tokens. In practice, with a 38% staking ratio and active governance, acquiring 0.1% of supply provides negligible voting power relative to the total. A realistic capture scenario requires acquiring a majority of all staked and locked tokens, which is economically infeasible for a mature network.

12.8.4 Voter Apathy and Quorum Design

Low voter turnout reduces governance security by lowering the absolute cost of achieving a quorum. QoreChain mitigates voter apathy through three mechanisms:

Validator Inheritance. Non-voting delegators inherit their validator's vote, ensuring that staked tokens are represented in governance even when delegators are passive. This effectively raises the participation floor to the validator participation rate.

Quorum Scaling. The quorum threshold q_τ is defined as a fraction of total voting power (not total supply), meaning it automatically adjusts as the staking ratio changes.

Participation Incentive. Governance participation contributes to validator reputation score r_i through the vote participation component $V_i(t)$, which in turn affects bonding curve rewards and QDRW voting power. Non-participating validators experience reputation decay:

$$r_i(t+1) = \lambda_R \cdot r_i(t) + (1 - \lambda_R) \cdot [\omega_u U_i(t) + \omega_p P_i(t) + \omega_v V_i(t)] \tag{12.55}$$

A validator who consistently abstains from governance ($V_i = 0$) experiences a steady-state reputation reduction, leading to lower block rewards through the bonding curve multiplier and lower governance influence through the reputation multiplier.

```
# Query governance participation statistics
qorechaind query gov participation-stats --output json
```

```
# Example response:
# {
#   "last_10_proposals_avg_turnout": "0.72",
#   "validator_participation_rate": "0.94",
#   "direct_voter_rate": "0.18",
#   "inherited_vote_rate": "0.54",
#   "governance_capture_cost_estimate": {
#     "supply_fraction_needed": "0.42",
#     "at_current_staking_ratio": "0.38"
#   }
# }
```

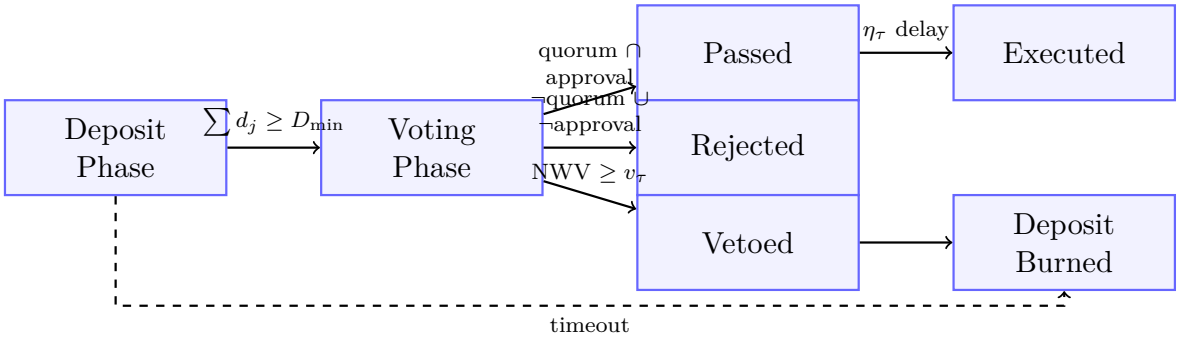

Figure 12.1: Proposal lifecycle state machine. Solid arrows indicate state transitions triggered by governance events; the dashed arrow indicates deposit timeout leading to deposit burn.

Chapter 13

Use Cases and Applications

QoreChain's convergence of post-quantum cryptography, AI-native intelligence, triple-VM execution, and multi-layer architecture creates a uniquely broad application surface. This chapter examines use cases across the full spectrum of network participants: from resource-constrained IoT devices to institutional-scale financial infrastructure, from individual community members to validator operators managing multi-source revenue streams.

Each use case is grounded in specific QoreChain capabilities documented in preceding chapters, with mathematical models quantifying the advantages and code examples demonstrating developer interaction patterns.

13.1 IoT and Lightweight Quantum-Safe Devices

The proliferation of connected devices creates an urgent need for quantum-safe identity and transaction signing on hardware with severe resource constraints. QoreChain addresses this through a lightweight PQC wallet designed for deployment on microcontrollers and edge gateways.

13.1.1 Lightweight PQC Wallet Architecture

The QoreChain IoT wallet implements ML-DSA-44 (a lower-security variant of the ML-DSA family) for constrained devices, with ML-DSA-87 available for devices with sufficient resources. The wallet operates in two modes:

Direct Mode. The device holds its own PQC keypair and signs transactions locally. Suitable for edge gateways, industrial controllers, and devices with ARM Cortex-A class processors or higher.

Delegated Mode. The device holds a lightweight attestation key and delegates transaction signing to a trusted gateway or light node. The device signs attestation payloads (sensor readings, status reports) which the gateway wraps in a full QoreChain transaction. Suitable for ARM Cortex-M class microcontrollers and sensor nodes.

13.1.2 Computational Cost Model

The signing cost on constrained hardware can be modelled as a function of the algorithm parameters and processor capability:

$$T_{\text{sign}}(d, f) = \frac{C_{\text{ops}}(d)}{f \cdot \text{IPC}} \tag{13.1}$$

where d is the ML-DSA security level (44, 65, or 87), f is the processor clock frequency, and IPC is the instructions-per-cycle ratio. For ML-DSA-44 on a representative set of embedded processors:

Table 13.1: ML-DSA-44 signing performance on constrained hardware

Processor Class	**Clock (MHz)**	T_{sign} (ms)	**RAM (KB)**
ARM Cortex-M4 (168 MHz)	168	∼85	∼48
ARM Cortex-M33 (100 MHz)	100	∼140	∼48
ARM Cortex-A53 (1.2 GHz)	1200	∼4.2	∼64
RISC-V (SiFive U74, 1.5 GHz)	1500	∼3.1	∼64

13.1.3 Key Size and Bandwidth Tradeoffs

PQC key and signature sizes are larger than classical ECDSA, creating bandwidth considerations for constrained networks:

$$B_{\text{tx}} = |pk| + |\sigma| + |m| + |\text{metadata}| \tag{13.2}$$

For ML-DSA-44: $|pk| = 1{,}312$ bytes, $|\sigma| = 2{,}420$ bytes, yielding a minimum transaction envelope of ∼3.8 KB compared to ∼0.1 KB for ECDSA. For low-bandwidth IoT networks (LoRaWAN, NB-IoT), the delegated mode amortises this overhead by batching multiple device attestations into a single gateway transaction:

$$B_{\text{amortised}} = \frac{|pk_{\text{gw}}| + |\sigma_{\text{gw}}| + \sum_{i=1}^{n}(|a_i| + |\sigma_i^{\text{attest}}|)}{n} \tag{13.3}$$

where a_i is the i-th device attestation payload and σ_i^{attest} is its lightweight attestation signature. For $n = 50$ batched attestations, the per-device bandwidth cost drops to ∼150 bytes.

13.1.4 Application Scenarios

Smart Grid Metering. Energy meters sign consumption readings with PQC attestation keys, creating quantum-safe audit trails for billing and regulatory compliance. A grid of 10,000 meters producing hourly readings generates ∼240,000 attestations per day, batched through ∼200 gateway transactions on QoreChain.

Autonomous Vehicle V2X. Vehicle-to-everything (V2X) communication requires tamper-proof, quantum-safe message authentication. QoreChain's PQC wallet enables vehicles to sign location, telemetry, and intent broadcasts that are verifiable on-chain for insurance, toll, and traffic management applications.

Industrial Supply Chain Provenance. Manufacturing sensors attest to production conditions (temperature, pressure, humidity) at each processing stage. The quantum-safe attestation chain provides immutable provenance records that remain verifiable post-quantum, critical for pharmaceutical, aerospace, and food safety supply chains.

Agricultural Monitoring. Soil sensors, weather stations, and drone-based imaging systems generate environmental data that is PQC-attested and anchored on-chain. Smart contracts on CosmWasm can trigger parametric crop insurance payouts (Section 13.3.6) based on verified sensor thresholds.

```
# Register an IoT gateway as a light node
qorechaind tx lightnode register \
  --node-type ux \
  --metadata '{"device_class": "gateway", "protocol": "lorawan"}' \
  --from gateway-key \
  --output json

# Submit a batched attestation transaction
qorechaind tx attestation submit-batch \
  --gateway qor1gateway...addr \
  --attestations batch_attestations.json \
  --from gateway-key \
  --output json

# Query device attestation history
qorechaind query attestation device-history \
  --device-id "meter-00421" \
  --limit 100 \
  --output json
```

13.2 Defense and Critical Infrastructure

National security and critical infrastructure applications demand the highest levels of cryptographic assurance, where the consequences of a key compromise extend far beyond financial loss. QoreChain's NIST FIPS 203/204/205 aligned cryptography positions it as infrastructure suitable for defense-adjacent and critical infrastructure use cases.

13.2.1 NSA CNSA 2.0 Compliance Pathway

The NSA Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) mandates migration to ML-KEM and ML-DSA for all national security systems by 2030. QoreChain's cryptographic layer natively implements both algorithms at their highest security levels (ML-DSA-87, ML-KEM-1024), satisfying the CNSA 2.0 requirements without requiring algorithmic migration:

$$\text{CNSA 2.0} \subseteq \text{QoreChain PQC Suite} = \{\text{ML-DSA-87}, \text{ML-KEM-1024}, \text{SHAKE-256}, \text{SLH-DSA}\} \tag{13.4}$$

The Algorithm Agility Framework (Chapter 4) enables governance-controlled addition of future algorithms as CNSA requirements evolve, without hard forks.

13.2.2 Classified Data Attestation

Defense applications require proof of data integrity without revealing content. QoreChain supports hash-on-chain attestation: a classified document's cryptographic hash is signed with ML-DSA-87 and recorded on-chain, providing quantum-safe proof of existence, integrity, and timestamp without any content disclosure:

$$\text{Attestation}(D) = \text{Sign}_{\text{ML-DSA-87}}(sk,\ H_{\text{SHAKE-256}}(D) \| t_{\text{block}} \| \text{metadata}) \tag{13.5}$$

The attestation is verifiable by any party holding the public key, while the document D remains entirely off-chain. The SHAKE-256 hash provides 256-bit quantum security for collision resistance.

13.2.3 Hidden Computation Sidechain for Defense Logistics

The HCS (Hidden Computation Sidechain) layer type, described in Chapter **??**, enables computation on private data with only PQC-signed state commitments anchored to the main chain. Defense logistics applications include:

Secure Supply Chain Coordination. Allied forces coordinate equipment and supply movements across a shared HCS, where each participant sees only the data relevant to their operational scope. The main chain anchor provides tamper-evident proof of coordination without exposing operational details.

C4ISR Data Integrity. Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance (C4ISR) systems generate vast quantities of sensitive data. HCS enables quantum-safe integrity verification of C4ISR data streams across organisational boundaries without centralised trust.

Coalition Interoperability. Cross-chain messaging with PQC attestation enables secure data sharing between allied blockchain systems. Each coalition partner operates an HCS instance, with cross-chain proofs enabling selective disclosure across trust boundaries:

$$\text{Proof}_{\text{cross}} = \text{Sign}_{\text{ML-DSA-87}}(sk_A,\ \text{MerkleProof}(d_A, R_A) \| \text{PolicyID}_B) \tag{13.6}$$

where d_A is the disclosed data element, R_A is the state root of Partner A's HCS, and PolicyID_B specifies Partner B's access scope.

13.2.4 Critical Infrastructure Protection

Power grids, water treatment systems, telecommunications networks, and transportation systems face increasing cyber threats. QoreChain provides quantum-safe infrastructure monitoring through:

Tamper-Evident Configuration Logs. Infrastructure control system configurations are hashed and attested on-chain. Any unauthorised modification is detectable by comparing current state against the on-chain commitment.

Quantum-Safe SCADA Authentication. Supervisory Control and Data Acquisition (SCADA) commands are signed with ML-DSA-87, preventing command injection attacks even from adversaries with quantum computing resources.

```
# Create an HCS layer for defense logistics
qorechaind tx multilayer register-sidechain \
  --name "Coalition Logistics" \
  --layer-type hcs \
  --min-validators 5 \
  --pqc-commitment-only true \
  --from admin-key \
  --output json

# Submit a classified document attestation
qorechaind tx attestation submit-hash \
  --document-hash "SHAKE256:a3f2...8b1c" \
  --classification-level "RESTRICTED" \
  --retention-years 50 \
  --from authorized-key \
  --output json
```

13.3 Financial Services and Institutional Applications

Financial services represent the sector most immediately impacted by the quantum threat. Every transaction, settlement record, and custody proof signed with classical cryptography today becomes vulnerable to retrospective forgery once cryptographically relevant quantum computers arrive. QoreChain provides a comprehensive quantum-safe financial infrastructure stack spanning settlement, custody, central bank integration, asset tokenization, trade finance, insurance, and institutional DeFi.

13.3.1 Quantum-Safe Settlement Infrastructure

Banks, exchanges, and clearinghouses require settlement finality with cryptographic guarantees that will remain valid for the lifetime of the settled obligations, often decades. QoreChain serves as a PQC-native settlement layer where every settlement proof is signed with ML-DSA-87 and verifiable against the quantum threat.

Real-Time Gross Settlement (RTGS). QoreChain's sub-second finality (designed target) enables real-time gross settlement of high-value transactions. Each settlement is atomic and final upon block commitment:

$$P(\text{settlement finality at block } h) = 1 - P(\text{reorg at depth } d \geq 1) \geq 1 - \left(\frac{f}{n}\right)^d \quad (13.7)$$

where $f < n/3$ under CPoS consensus. At $d = 1$ (immediate finality), with $f/n = 0.1$: $P(\text{finality}) \geq 0.9$, improving to ≥ 0.99 at $d = 2$.

CNSA 2.0 Financial Compliance. Financial institutions operating under NSA CNSA 2.0 timelines and European quantum readiness directives (ENISA, EBA) require settlement infrastructure that satisfies post-quantum mandates. QoreChain's

native PQC implementation eliminates the migration burden: institutions settling on QoreChain are compliant from day one.

Counterparty Risk Reduction. The quantum threat introduces a novel form of counterparty risk: a counterparty's historical transaction signatures may be forged post-quantum, enabling repudiation of past obligations. QoreChain eliminates this risk class entirely:

$$\text{CounterpartyRisk}_{\text{PQC}} = \text{CounterpartyRisk}_{\text{classical}} - \underbrace{P(\text{quantum forgery}) \cdot \text{ExposureValue}}_{\text{eliminated by PQC}} \tag{13.8}$$

For a portfolio with 20-year obligation horizon and current quantum threat probability estimates ($P \approx 0.15$–0.30 by 2035):

$$\Delta\text{Risk} = P(\text{CRQC}_{2035}) \cdot \sum_i \text{NPV}(\text{Obligation}_i) \cdot P(\text{forgery}_i|\text{CRQC}) \tag{13.9}$$

```
# Execute a quantum-safe settlement transaction
qorechaind tx bank send \
  qor1bank...sender qor1clearing...receiver \
  500000000000000uqor \
  --note "RTGS Settlement #2026-03-22-0041" \
  --pqc-attestation required \
  --from bank-settlement-key \
  --output json

# Query settlement finality proof
qorechaind query tx settlement-proof \
  --tx-hash "A4F2...8B1C" \
  --include-pqc-signature true \
  --output json

# Example response:
# {
#   "tx_hash": "A4F2...8B1C",
#   "block_height": 5240100,
#   "finality_depth": 1,
#   "pqc_algorithm": "ML-DSA-87",
#   "signature_valid": true,
#   "settlement_time_ms": 842,
#   "counterparty_proof": {
#     "sender_pqc_pubkey": "qorpub1...",
#     "receiver_pqc_pubkey": "qorpub1...",
#     "merkle_inclusion_proof": "..."
#   }
# }
```

13.3.2 Institutional Digital Asset Custody

Institutional custody of digital assets requires key management infrastructure that will remain secure for the duration of asset holding periods, which may span decades for endowments, pension funds, and sovereign wealth funds. QoreChain provides a PQC-native custody framework.

PQC Key Encapsulation for Key Rotation. ML-KEM-1024 key encapsulation enables secure key rotation for custodial wallets. When rotating from keypair $(pk_{\text{old}}, sk_{\text{old}})$ to $(pk_{\text{new}}, sk_{\text{new}})$, the new public key is encapsulated and distributed through a quantum-safe channel:

$$(c, K) \leftarrow \text{Encaps}_{\text{ML-KEM-1024}}(pk_{\text{new}}), \qquad K' \leftarrow \text{Decaps}_{\text{ML-KEM-1024}}(sk_{\text{new}}, c) \tag{13.10}$$

The shared secret $K = K'$ is used to derive a session key for secure transfer of custody rights. The encapsulation ciphertext c can be stored on-chain as an auditable rotation record.

Abstract Account Spending Rules. The `x/abstractaccount` module enables institutional policy enforcement through programmable spending rules and session keys:

- **Daily transfer limits**: Maximum outbound value per 24-hour rolling window.
- **Multi-approval thresholds**: Transfers above a configurable threshold require k-of-n PQC signatures from authorised signers.
- **Time-locked transfers**: Large transfers are subject to a mandatory delay period during which they can be cancelled by a quorum of custodians.
- **Session keys**: Temporary keys with scoped permissions for operational use, automatically expiring after a configured block height or time duration.
- **Whitelisted recipients**: Transfers restricted to a pre-approved set of destination addresses.

Multi-Party PQC Threshold Signature Model. For institutional custody requiring distributed key management, a k-of-n threshold scheme over ML-DSA-87 provides:

$$\text{Security}(k, n) = 1 - \binom{n}{k-1}^{-1} \cdot P(\text{compromise per signer})^k \tag{13.11}$$

For a 3-of-5 custodial setup with per-signer compromise probability $p = 10^{-4}$:

$$P(\text{custody breach}) = \binom{5}{3} \cdot (10^{-4})^3 = 10 \cdot 10^{-12} = 10^{-11} \tag{13.12}$$

Combined with ML-DSA-87's 2^{128} quantum security margin, the total custody security is dominated by operational security rather than cryptographic risk.

```
# Create an abstract account with institutional spending rules
qorechaind tx abstractaccount create \
  --spending-rules '{
    "daily_limit": "100000000000uqor",
    "multi_approval_threshold": "50000000000uqor",
    "required_signers": 3,
    "total_signers": 5,
    "time_lock_blocks": 1200,
    "whitelisted_recipients": [
      "qor1clearing...addr",
      "qor1treasury...addr"
    ]
  }' \
  --session-key-ttl 14400 \
  --from custodian-master-key \
  --output json

# Query abstract account policy
qorechaind query abstractaccount policy qor1custody...addr --output json
```

13.3.3 Central Bank Digital Currency (CBDC) Rails

Central banks worldwide are developing digital currencies that require settlement infrastructure meeting the highest standards of cryptographic assurance, throughput, and regulatory control. QoreChain provides three CBDC deployment models, each leveraging different architectural layers.

Model 1: Wholesale CBDC on Enterprise Rollup. A central bank deploys a sovereign RDK rollup (Enterprise profile) as its CBDC ledger, with state anchored to the QoreChain main chain. The rollup provides the central bank with full control over monetary policy, account management, and compliance rules, while the main chain anchor provides quantum-safe settlement finality and cross-border interoperability:

$$\text{CBDC}_{\text{wholesale}} \xrightarrow{\text{state anchor}} \text{QoreChain Main Chain} \xrightarrow{\text{One-Hop Swap}} \text{Partner CBDC} \tag{13.13}$$

Model 2: Retail CBDC on Paychain. For high-frequency retail transactions, a central bank registers a paychain layer optimised for micropayments with sub-500ms target block time and up to 50 active paychains. The paychain settles to the main chain at configurable intervals, providing retail-speed transactions with main-chain security guarantees.

Model 3: Cross-Border CBDC Corridors. Multiple central banks each operate sovereign rollups or paychains, connected through QoreChain's One-Hop-Swap topology. A cross-border CBDC transfer follows:

$$\text{CBDC}_A \xrightarrow{\text{Bridge}_A} \text{QoreChain} \xrightarrow{\text{Bridge}_B} \text{CBDC}_B \tag{13.14}$$

The QCAI routing engine selects the optimal bridge protocol (IBC or QCB) based on fee, speed, and security parameters, enabling interoperable CBDC corridors without bilateral integration between central banks.

Settlement Throughput Requirements. A national retail CBDC system serving a population of 50 million with an average of 3 transactions per person per day requires:

$$\text{TPS}_{\text{required}} = \frac{50 \times 10^6 \times 3}{86{,}400} \approx 1{,}736 \text{ TPS} \tag{13.15}$$

This falls well within QoreChain's designed throughput of 5,000+ TPS on the main chain, with paychain layers providing additional horizontal scaling for peak demand periods.

```
# Deploy a wholesale CBDC rollup (Enterprise profile)
qorechaind tx rdk create-rollup \
  --name "National Digital Currency" \
  --profile enterprise \
  --settlement-mode zk \
  --vm-type evm \
  --sequencer-mode dedicated \
  --stake 100000000000uqor \
  --from central-bank-key \
  --output json

# Register a retail CBDC paychain
```

```
qorechaind tx multilayer register-paychain \
  --name "Retail CBDC Payments" \
  --target-block-time 500ms \
  --min-stake 100000000uqor \
  --settlement-interval 100 \
  --from central-bank-key \
  --output json
```

13.3.4 Securities Tokenization and Real-World Assets

Tokenization of real-world assets (bonds, equities, real estate, commodities) creates digital representations whose ownership proofs must remain cryptographically valid for the lifetime of the underlying asset. Classical ECDSA signatures on tokenized assets face a unique risk: if quantum computers break ECDSA before the asset matures, historical ownership transfers can be forged, creating legal disputes over provenance.

PQC-Signed Ownership Proofs. Every token transfer on QoreChain produces an ML-DSA-87 signed ownership proof that remains valid against quantum attacks:

$$\text{OwnershipProof}(a, t) = \text{Sign}_{\text{ML-DSA-87}}(sk_{\text{issuer}},\ H(a \| \text{owner}_t \| t_{\text{block}} \| \text{terms})) \tag{13.16}$$

For a 30-year government bond tokenized in 2027, the ownership proof remains verifiable through 2057, well beyond the most conservative quantum threat timelines.

On-Chain Corporate Actions. Tokenized securities enable automated corporate actions executed by smart contracts:

- **Dividend distribution**: Pro-rata distribution to all token holders at a snapshot block height, with quantum-safe proof of entitlement.
- **Shareholder voting**: QDRW governance applied to shareholder votes, with voting power proportional to holdings and dampened by square root to prevent single-shareholder dominance.
- **Stock splits and consolidations**: Atomic on-chain operations affecting all holders simultaneously, with PQC-signed audit trail.

Settlement Cost Reduction. Traditional securities settlement operates on a T+2 cycle with multiple intermediaries. QoreChain enables T+0 atomic settlement:

$$\text{Cost}_{\text{traditional}} = c_{\text{broker}} + c_{\text{clearing}} + c_{\text{CSD}} + c_{\text{custodian}} + c_{\text{settlement}} + c_{\text{reconciliation}} \tag{13.17}$$

$$\text{Cost}_{\text{QoreChain}} = c_{\text{gas}} + c_{\text{bridge}} \approx 0.01 \cdot \text{Cost}_{\text{traditional}} \tag{13.18}$$

The elimination of intermediary reconciliation, counterparty risk margins, and multi-day settlement float produces substantial cost savings at institutional scale. For a market processing 1 million settlement transactions daily at an average settlement cost of \$5 per transaction:

$$\text{Annual Savings} = 365 \times 10^6 \times \$5 \times 0.99 \approx \$1.8 \text{ billion} \tag{13.19}$$

Compliance via HCS Selective Disclosure. Regulated securities require compliance checks (KYC/AML, accredited investor verification, jurisdictional restrictions) without exposing personal data on a public ledger. The HCS layer enables selective disclosure: compliance proofs are computed off-chain and only PQC-signed attestation commitments are anchored on the main chain.

```
# Deploy a tokenized bond contract (EVM)
qorechaind tx evm create \
  --contract bond_token.sol \
  --args '["30Y-GOV-BOND-2027", "BOND30", 1000000, 1735689600]' \
  --from issuer-key \
  --output json

# Execute a T+0 atomic settlement
qorechaind tx evm call \
  --contract qor1bond...contract \
  --method "atomicSettle(address,address,uint256)" \
  --args '["qor1seller...addr", "qor1buyer...addr", 50000]' \
  --from settlement-engine-key \
  --output json

# Query ownership proof for audit
qorechaind query evm contract-state \
  --contract qor1bond...contract \
  --method "getOwnershipProof(address)" \
  --args '["qor1holder...addr"]' \
  --output json
```

13.3.5 Trade Finance and Cross-Border Payments

International trade involves complex multi-party workflows (letters of credit, bills of lading, certificates of origin) that currently rely on paper documents and multiple intermediaries. QoreChain digitises these workflows with quantum-safe guarantees and cross-chain settlement.

Letters of Credit on CosmWasm. A letter of credit (LC) is encoded as a CosmWasm smart contract that automatically releases payment upon verified delivery of goods. The LC contract enforces:

1. Buyer deposits funds into escrow (PQC-signed).
2. Shipping company attests delivery (PQC-signed attestation).
3. Customs authority attests clearance (PQC-signed attestation).
4. Contract verifies all attestations and releases escrow to seller.

Each attestation is signed with ML-DSA-87, ensuring that the trade finance record remains legally valid against quantum-era forgery challenges.

Cross-Border Payment Corridors via One-Hop-Swap. International remittances currently incur 5–7% fees through traditional corridors. QoreChain's One-Hop-Swap topology enables any-to-any currency transfer through a single intermediate hop:

$$\text{Currency}_A \xrightarrow{\text{Bridge}_A} \text{QOR} \xrightarrow{\text{Bridge}_B} \text{Currency}_B \tag{13.20}$$

The total remittance cost under the One-Hop-Swap model:

$$C_{\text{remit}} = \phi_{\text{in}} + \phi_{\text{swap}} + \phi_{\text{out}} + \delta_{\text{slippage}} \tag{13.21}$$

where ϕ_{in} and ϕ_{out} are bridge fees, ϕ_{swap} is the on-chain swap fee, and δ_{slippage} is the price impact. For a \$1,000 remittance with typical bridge fees of 0.1% and swap fees of 0.3%:

$$C_{\text{remit}} = \$1.00 + \$3.00 + \$1.00 + \$0.50 = \$5.50 \quad (0.55\%) \tag{13.22}$$

This represents a 90% reduction compared to the traditional 5–7% corridor, with settlement in minutes rather than days.

Gas Abstraction for Multi-Currency Operations. The `x/gasabstraction` module enables trade finance participants to pay transaction fees in IBC-bridged stablecoins (USDC, USDT) or other tokens, eliminating the requirement to hold QOR for gas. The abstraction layer converts non-native denominations at configurable exchange rates:

$$\text{GasCost}_{\text{native}} = \text{GasCost}_{\text{uqor}} \cdot r_{\text{denom}} \tag{13.23}$$

where r_{denom} is the governance-set conversion rate for the payment denomination.

```
# Deploy a letter of credit contract (CosmWasm)
qorechaind tx wasm instantiate \
  --code-id 42 \
  --label "LC-2026-TRADE-0088" \
  --msg '{
    "buyer": "qor1buyer...addr",
    "seller": "qor1seller...addr",
    "amount": "500000000000uqor",
    "shipping_attestor": "qor1shipping...addr",
    "customs_attestor": "qor1customs...addr",
    "expiry_blocks": 100800
  }' \
  --funds 500000000000uqor \
  --from buyer-key \
  --output json

# Execute a cross-border remittance via One-Hop-Swap
qorechaind tx bridge transfer \
  --source-chain ethereum \
  --dest-chain solana \
  --amount 1000000000 \
  --denom ibc/USDC \
  --recipient "SolAddr...xyz" \
  --route auto \
  --output json
```

13.3.6 Insurance and Parametric Contracts

Parametric insurance pays out automatically when predefined conditions are met, eliminating claims disputes and reducing administrative overhead. QoreChain enables parametric insurance contracts that leverage both on-chain data and QCAI-verified external events.

Parametric Trigger Model. A parametric insurance contract defines a trigger function $\tau(\mathbf{x})$ over a vector of observed parameters $\mathbf{x}$:

$$\text{Payout}(\mathbf{x}) = \begin{cases} P_{\max} \cdot f(\mathbf{x}) & \text{if } \tau(\mathbf{x}) = 1 \\ 0 & \text{if } \tau(\mathbf{x}) = 0 \end{cases} \tag{13.24}$$

where $f(\mathbf{x})$ is a severity function mapping observed parameters to payout fraction. For example, a crop insurance contract with rainfall trigger:

$$\tau(\text{rainfall}) = \mathbf{1}[\text{rainfall} < \text{threshold}], \qquad f(\text{rainfall}) = \min\left(1, \frac{\text{threshold} - \text{rainfall}}{\text{threshold} - \text{min\_rainfall}}\right) \tag{13.25}$$

QCAI Event Verification. For on-chain data (IoT sensor readings attested per Section 13.1), QCAI provides anomaly detection to verify that trigger events are genuine rather than the result of sensor malfunction or data manipulation. The verification pipeline applies:

$$P(\text{genuine event} \mid \mathbf{x}) = \frac{P(\mathbf{x} \mid \text{genuine}) \cdot P(\text{genuine})}{P(\mathbf{x} \mid \text{genuine}) \cdot P(\text{genuine}) + P(\mathbf{x} \mid \text{anomaly}) \cdot P(\text{anomaly})} \tag{13.26}$$

The contract triggers payout only when $P(\text{genuine event} \mid \mathbf{x}) \geq \theta_{\text{confidence}}$ (default: 0.95).

Reinsurance Syndication. Multiple insurers can pool risk across QoreChain's cross-chain infrastructure. A reinsurance syndicate contract distributes premium income and claims obligations proportionally among participants, with settlement occurring on the main chain and individual policy contracts running on rollups:

$$\text{ClaimShare}_i = \frac{s_i}{\sum_j s_j} \cdot \text{TotalClaim}, \qquad \text{PremiumShare}_i = \frac{s_i}{\sum_j s_j} \cdot \text{TotalPremium} \tag{13.27}$$

where s_i is insurer i's syndicate stake.

```
# Deploy a parametric crop insurance contract
qorechaind tx wasm instantiate \
  --code-id 55 \
  --label "CropInsurance-Region-42" \
  --msg '{
    "insurer": "qor1insurer...addr",
    "policyholder": "qor1farmer...addr",
    "trigger_type": "rainfall_deficit",
    "threshold_mm": 200,
    "min_rainfall_mm": 50,
    "max_payout": "10000000000uqor",
```

```
  "data_source": "qor1weather...oracle",
  "confidence_threshold": "0.95",
  "coverage_period_blocks": 525600
}' \
--funds 10000000000uqor \
--from insurer-key \
--output json
```

13.3.7 Institutional DeFi

Institutional participation in decentralised finance requires regulatory compliance, quantum-safe security, and enterprise-grade infrastructure that consumer-oriented DeFi protocols do not provide. QoreChain bridges this gap through permissioned execution environments and institutional-grade financial primitives.

Permissioned DeFi on Enterprise Rollups. The RDK Enterprise rollup profile (Chapter **??**) provides a permissioned execution environment where participation is restricted to KYC-verified entities. DeFi protocols deployed on an Enterprise rollup inherit the main chain's PQC security while enforcing compliance at the application layer:

$$\text{Access}(a, \text{rollup}_E) = \text{KYC}(a) \wedge \text{AML}(a) \wedge \text{Jurisdiction}(a) \wedge \text{Accreditation}(a) \quad (13.28)$$

Quantum-Safe Lending and Borrowing. Lending protocols on QoreChain use PQC-signed collateral proofs that remain valid post-quantum. The collateral proof includes a Merkle inclusion proof of the collateral position within the state tree, signed with ML-DSA-87:

$$\text{CollateralProof}(c, t) = \text{Sign}_{\text{ML-DSA-87}}(sk_{\text{protocol}},\ \text{MerkleProof}(c, R_t) \| V(c) \| \text{LTV}) \quad (13.29)$$

where $V(c)$ is the collateral valuation at time t and LTV is the loan-to-value ratio. This proof can be presented to external auditors or regulators as quantum-safe evidence of protocol solvency.

Institutional Yield via xQORE. Institutions seeking yield with governance exposure can lock QOR into xQORE (Section 11.5), receiving enhanced governance voting power (2x efficiency) and PvP rebase yield from early exits by other participants. The institutional xQORE yield is predictable under steady-state conditions:

$$\text{APY}_{\text{institutional}} = \text{APY}_{\text{staking}} + \text{APY}_{\text{PvP}} + \text{APY}_{\text{governance\_}value} \quad (13.30)$$

where $\text{APY}_{\text{governance\_}value}$ represents the economic value of enhanced governance influence, quantifiable as the cost of acquiring equivalent voting power through additional token purchase.

Treasury Management with QDRW Governance. Institutional treasuries holding QOR benefit from the QDRW governance model, which provides meaningful governance influence even for mid-size holders through the square-root dampening function. An institution with 0.5% of staked supply obtains governance influence equivalent to:

$$\text{VP}_{\text{institution}} = \sqrt{0.005 \cdot S_{\text{total}} \cdot Q(r)} \tag{13.31}$$

With strong reputation ($Q = 2.0$), the institution's effective governance weight exceeds its raw capital share, incentivising long-term, responsible participation.

```
# Deploy a KYC-gated lending protocol on Enterprise rollup
qorechaind tx rdk deploy-contract \
  --rollup-id "enterprise-layer-01" \
  --contract lending_protocol.sol \
  --kyc-required true \
  --jurisdiction-whitelist '["CH", "SG", "AE", "HK"]' \
  --from institutional-deployer-key \
  --output json

# Lock QOR into xQORE for institutional yield
qorechaind tx xqore lock \
  --amount 5000000000000000uqor \
  --lock-duration 180 \
  --from treasury-key \
  --output json

# Query institutional portfolio summary
qorechaind query portfolio institutional-summary \
  --address qor1treasury...addr \
  --output json

# Example response:
# {
#   "staked_qor": "3000000000000000",
#   "xqore_locked": "5000000000000000",
#   "lock_remaining_days": 147,
#   "voting_power": "28284.27",
#   "voting_power_pct": "0.0075",
#   "estimated_apy": {
#     "staking": "0.142",
#     "pvp_rebase": "0.018",
#     "total": "0.160"
#   },
#   "governance_proposals_voted": 12,
#   "reputation_score": "0.91"
# }
```

13.4 Developer Ecosystem: QoreChain Studio

QoreChain Studio is the AI-powered development environment for building, testing, auditing, and deploying smart contracts across all three virtual machines (EVM, CosmWasm, SVM). Powered by QCAI, the Studio provides natural-language contract generation, automated security auditing, and cross-VM deployment tooling.

13.4.1 QCAI Contract Generation

Developers describe contract requirements in natural language, and QCAI generates audited smart contract code in the target VM language (Solidity, Rust/CosmWasm, or Rust/SVM):

1. **Requirement Analysis**: QCAI parses the natural-language specification, identifying contract type, state variables, access control patterns, and external dependencies.
2. **Code Generation**: QCAI produces a complete contract implementation with PQC-aware patterns (using `qor_pqc_verify` precompiles where applicable).
3. **Automated Audit**: The generated contract is immediately passed through the QCAI audit pipeline (Section 13.4.2) before delivery to the developer.
4. **Test Suite Generation**: QCAI generates a companion test suite covering standard paths, edge cases, and adversarial inputs.

The generation pipeline supports three QCAI tiers:

Table 13.2: QCAI contract generation tiers

QCAI Tier	Use Case	Latency	Audit Depth
QCAI Fast	Simple tokens, basic logic	<5 s	Surface-level
QCAI Balanced	DeFi protocols, multi-contract	<30 s	Standard
QCAI Advanced	Complex cross-VM, formal hints	<120 s	Deep analysis

13.4.2 QCAI Deep Contract Audit

The QCAI audit engine performs multi-layer security analysis on smart contracts, identifying vulnerabilities across categories:

Vulnerability Classes. Reentrancy, integer overflow/underflow, access control violations, unchecked external calls, front-running susceptibility, oracle manipulation, flash-loan attack vectors, gas griefing, storage collision, and PQC signature verification bypass.

Audit Scoring. Each contract receives a composite security score:

$$\mathcal{Q}(\mathcal{C}) = 1 - \max_{v \in \mathcal{V}(\mathcal{C})} \text{Severity}(v) - \lambda_a \cdot \frac{|\mathcal{V}(\mathcal{C})|}{|\mathcal{L}(\mathcal{C})|} \tag{13.32}$$

where $\mathcal{V}(\mathcal{C})$ is the set of detected vulnerabilities, $|\mathcal{L}(\mathcal{C})|$ is the number of code lines, and λ_a is the vulnerability density penalty coefficient. Contracts scoring below 0.7 receive a "Review Required" flag; below 0.5 receives "Deploy Not Recommended".

Gas Optimisation. QCAI identifies gas-inefficient patterns and suggests optimisations, estimating the gas reduction for each recommendation:

$$\Delta G = G_{\text{original}} - G_{\text{optimised}} = \sum_i \delta g_i \cdot n_i \tag{13.33}$$

where δg_i is the gas saving per occurrence of pattern i and n_i is the occurrence count.

13.4.3 Cross-VM Development

QoreChain Studio provides a unified development experience across all three VMs. A developer writing an EVM contract can invoke a CosmWasm module or trigger an SVM program through the cross-VM messaging layer, all within a single atomic transaction:

```
// Solidity contract calling a CosmWasm module via cross-VM
pragma solidity ^0.8.20;

import "@qorechain/crossvm/ICosmWasmBridge.sol";

contract CrossVMDeFi {
    ICosmWasmBridge public bridge;

    constructor(address _bridge){
        bridge = ICosmWasmBridge(_bridge);
    }

    // Invoke a CosmWasm lending pool from EVM
    function depositToCosmWasmPool(
        string calldata poolContract,
        uint256 amount
    ) external {
        bytes memory msg_payload = abi.encodePacked(
            '{"deposit":{"amount":"', amount, '"'}}
        );
        bridge.executeCosmWasm(poolContract, msg_payload);
    }
}
```

```
# Generate a contract using QCAI
qorechaind tx ai generate-contract \
  --description "ERC-20 token with PQC-verified minting,
    100M max supply, admin-controlled pause, and quantum-safe
    ownership transfer" \
  --vm evm \
  --tier balanced \
  --audit true \
  --from developer-key \
  --output json

# Audit an existing contract
qorechaind tx ai audit-contract \
  --contract-path ./MyProtocol.sol \
  --tier advanced \
  --include-gas-optimization true \
  --from developer-key \
  --output json

# Example audit response:
# {
```

```
#   "contract": "MyProtocol.sol",
#   "security_score": 0.82,
#   "vulnerabilities": [
#     {
#       "type": "unchecked_return_value",
#       "severity": "medium",
#       "line": 142,
#       "recommendation": "Check return value of transfer()"
#     }
#   ],
#   "gas_optimizations": [
#     {
#       "pattern": "redundant_sload",
#       "occurrences": 3,
#       "estimated_saving_gas": 2400,
#       "suggestion": "Cache storage variable in memory"
#     }
#   ],
#   "pqc_compliance": "pass"
# }
```

13.5 Multi-Revenue Validator Operations

QoreChain validators operate as multi-dimensional economic actors, earning revenue from multiple independent sources through a single validator infrastructure. This section quantifies the composite revenue model and analyses return on investment across income channels.

13.5.1 Revenue Streams

A fully licensed QoreChain validator earns from six distinct sources:

1. Block Rewards (37% Fee Share + Bonding Curve). Validators receive 37% of all transaction fees, distributed proportionally to their selection probability. The bonding curve (Section 10.4) provides additional rewards based on stake, longevity, and reputation:

$$R_{\text{block}}(v) = 0.37 \cdot F_{\text{block}} \cdot P_{\text{select}}(v) + R_{\text{bonding}}(v, t) \tag{13.34}$$

2. Bridge Attestation Fees. Licensed bridge validators earn per-attestation fees for cross-chain event verification. Each attestation of a deposit, withdrawal, or relay event on a licensed chain generates a fee:

$$R_{\text{bridge}}(v) = \sum_{c \in \mathcal{C}_v^{\text{licensed}}} n_c \cdot \phi_c \tag{13.35}$$

where $\mathcal{C}_v^{\text{licensed}}$ is the set of chains for which validator v holds active bridge licenses, n_c is the number of attestations on chain c, and ϕ_c is the per-attestation fee for chain c.

3. Delegator Commissions. Validators set a commission rate $\kappa_v \in [0.05, 0.20]$ on delegator rewards:

$$R_{\text{commission}}(v) = \kappa_v \cdot \sum_{j \in \mathcal{D}_v} R_{\text{delegation}}(j) \tag{13.36}$$

4. xQORE Staking Yield. Validators who lock their self-bonded stake into xQORE earn PvP rebase yield (Section 11.5) in addition to standard staking returns.

5. Rollup Sequencer Fees. Validators serving as dedicated sequencers for RDK rollups earn sequencer fees from rollup transaction processing:

$$R_{\text{sequencer}}(v) = \sum_{r \in \mathcal{R}_v} \phi_r \cdot \text{TxCount}_r \tag{13.37}$$

6. Light Node Delegation Overflow. When light nodes delegate to a validator, the validator's total selection weight increases, generating additional block rewards proportional to the delegated light node stake.

13.5.2 Composite Revenue Model

The composite revenue model reflects QoreChain's multi-capability architecture and creates diversified income streams that allow validators to optimize their participation strategy based on capital constraints and operational capabilities. Rather than a single block reward model, validators can choose to specialize in different value-creation activities: consensus participation (block rewards), cross-chain services (bridge revenues), delegation incentives (commission), AI service provision (QCAI fees), transaction ordering (sequencer fees), and light client support (light node fees). This specialization allows validators with different hardware and network capabilities to find profitable niches. A validator with high-speed networking might optimize for sequencing, while a validator with specialized GPU hardware might focus on QCAI services. This economic flexibility increases the total number of viable validator configurations, improving network resilience by reducing the penalty for hardware heterogeneity.

Total annual revenue for a fully licensed validator:

$$R_{\text{total}}(v) = R_{\text{block}}(v) + R_{\text{bridge}}(v) + R_{\text{commission}}(v) + R_{\text{xqore}}(v) + R_{\text{sequencer}}(v) + R_{\text{lightnode}}(v) \tag{13.38}$$

13.5.3 License-Gated Bridge Expansion

The `x/license` module governs bridge validator operations through 27 feature IDs covering all supported chains. Each license grants the validator authority to run bridge watcher services for the specified chain and earn attestation fees:

Each additional bridge license increases the validator's revenue surface by adding a new attestation fee stream. The marginal ROI of adding a bridge license for chain c:

$$\text{ROI}_c = \frac{R_{\text{bridge},c} - C_{\text{infra},c}}{C_{\text{infra},c}} = \frac{n_c \cdot \phi_c - (C_{\text{server}} + C_{\text{rpc}} + C_{\text{license}})}{C_{\text{server}} + C_{\text{rpc}} + C_{\text{license}}} \tag{13.39}$$

where $C_{\text{infra},c}$ includes the cost of running the chain-specific watcher service (Docker container), RPC endpoint access for the target chain, and the license acquisition cost.

Table 13.3: Bridge license categories (representative subset)

Feature ID	Chain	Type
FeatureBridgeEthereum	Ethereum	QCB Bridge
FeatureBridgeSolana	Solana	QCB Bridge
FeatureBridgeBSC	BNB Smart Chain	QCB Bridge
FeatureValidatorEthereum	Ethereum	Validator
FeatureValidatorSolana	Solana	Validator
FeatureQCBBridge	QCB Protocol	Global

13.5.4 ROI Analysis

A validator's overall return on investment:

$$\text{ROI}_{\text{total}} = \frac{R_{\text{total}}(v) - C_{\text{total}}(v)}{C_{\text{total}}(v)} \tag{13.40}$$

where total costs include:

$$C_{\text{total}}(v) = C_{\text{hardware}} + C_{\text{bandwidth}} + C_{\text{stake\_opportunity}} + \sum_{c} C_{\text{infra},c} + C_{\text{operations}} \tag{13.41}$$

The multi-revenue model provides revenue diversification: even if bridge volumes decline on one chain, the validator continues earning from block rewards, commissions, and sequencer fees. The Sharpe ratio of the diversified validator revenue stream:

$$\text{SR}_{\text{validator}} = \frac{\mathbb{E}[R_{\text{total}}] - r_f}{\sigma(R_{\text{total}})} > \frac{\mathbb{E}[R_{\text{block}}] - r_f}{\sigma(R_{\text{block}})} = \text{SR}_{\text{block}} - only \tag{13.42}$$

since uncorrelated revenue streams reduce portfolio volatility while maintaining or increasing expected returns.

```
# Query validator revenue breakdown
qorechaind query validator revenue qorvaloper1abc...def --output json

# Example response:
# {
#   "validator": "qorvaloper1abc...def",
#   "period": "30d",
#   "revenue": {
#     "block_rewards": "45200000000uqor",
#     "bonding_curve": "12800000000uqor",
#     "bridge_attestations": "18500000000uqor",
#     "commissions": "8900000000uqor",
#     "xqore_rebase": "3200000000uqor",
#     "sequencer_fees": "6100000000uqor",
#     "total": "94700000000uqor"
#   },
#   "active_licenses": [
#     "FeatureBridgeEthereum",
#     "FeatureBridgeSolana",
```

```
#     "FeatureBridgeBSC",
#     "FeatureValidatorEthereum",
#     "FeatureQCBBridge"
#   ],
#   "delegators": 247,
#   "commission_rate": "0.10"
# }

# Acquire a bridge license for a new chain
qorechaind tx license grant \
  --grantee qorvaloper1abc...def \
  --feature-id FeatureBridgeAvalanche \
  --duration 365d \
  --from admin-key \
  --output json
```

13.6 Community and Consumer Applications

QoreChain is designed for participation at every level, from passive token holders to active community contributors. This section covers use cases accessible to individual users without institutional infrastructure.

13.6.1 Light Node Participation

Light nodes (Chapter 8) enable any user with standard hardware (laptop, desktop, or home server) to participate in network validation and earn the dedicated 3% share of all transaction fees. Two editions are available:

SX (Server eXperience). A persistent daemon process for servers and always-on devices. Provides full light node services including transaction relay, state validation, and network monitoring. Minimum requirements: 100 QOR delegated stake, 80% uptime.

UX (User eXperience). An embedded web dashboard with a lightweight daemon, enabling browser-based participation and monitoring. Designed for non-technical users who want to contribute to network decentralisation while earning passive income.

Light node rewards are distributed proportionally to delegated stake and uptime:

$$R_{\text{ln}}(i) = \frac{w_i \cdot u_i}{\sum_j w_j \cdot u_j} \cdot 0.03 \cdot F_{\text{total}} \tag{13.43}$$

where w_i is delegated stake weight and u_i is the uptime ratio (must exceed 80% to qualify).

13.6.2 Quantum-Safe Personal Wallet and Identity

Every QoreChain account is natively protected by ML-DSA-87 signatures, providing quantum-safe personal asset security without any additional configuration. Users benefit from:

Future-Proof Asset Protection. Assets held in QoreChain wallets are signed with PQC keys, ensuring that even if classical cryptography is broken by quantum

computers, the user's assets remain secure. Wallet recovery mechanisms use threshold secret sharing to distribute key backup material across multiple secure locations, eliminating single points of failure in key management while preserving the quantum-safe security properties of the underlying cryptography.

Quantum-Safe Identity Attestation. Users can create on-chain identity attestations (verified email, social accounts, organisational membership) signed with ML-DSA-87. These attestations serve as portable, quantum-safe digital identity anchors across the QoreChain ecosystem. Attestation verifiers can validate credentials against trusted issuers without requiring access to off-chain identity databases, enabling decentralised identity verification for financial services, regulatory compliance, and institutional partnerships.

Privacy-Preserving Account Abstraction. The wallet layer supports account abstraction patterns that improve user experience by enabling sponsored transactions, batch operations, and social recovery without compromising PQC security. Wallet deployments can operate with local signing (user controls keys) or delegate signing to hardware security modules (HSMs) that maintain PQC key material in tamper-resistant enclaves, protecting against key theft while maintaining user custody.

Multi-Signature and Threshold Governance. Personal wallets support multi-signature schemes where m of n signatures are required to authorize transactions. This enables family wealth management, corporate treasuries, and DAO treasuries to implement checks and balances. The multi-signature scheme uses secure aggregate signing protocols to minimize on-chain footprint compared to naive multi-signature concatenation, reducing transaction costs for governance-heavy operations.

13.6.3 Cross-Chain Asset Management via One-Hop-Swap

Individual users benefit from QoreChain's hub topology for cross-chain transfers. Any token on any of the 25 connected chains can be transferred to any other connected chain through a maximum of two hops (source chain to QoreChain, QoreChain to destination chain):

$$\text{MaxHops}(A, B) = \begin{cases} 1 & \text{if } A = \text{QoreChain or } B = \text{QoreChain} \\ 2 & \text{otherwise} \end{cases} \tag{13.44}$$

QCAI automatically selects the optimal route (IBC or QCB) based on user preferences (minimise fee, minimise time, or maximise security), presenting a simple interface that abstracts the underlying bridge complexity.

13.6.4 Governance Participation with QDRW

Small token holders are specifically empowered by the QDRW governance model. A user holding just 1,000 QOR and maintaining good reputation ($r = 0.8$, $Q = 1.7$) obtains voting power:

$$VP = \sqrt{1{,}000} \cdot 1.7 = 31.62 \cdot 1.7 = 53.8 \tag{13.45}$$

Compare with a whale holding 1,000,000 QOR with average reputation ($r = 0.5$, $Q = 1.25$):

$$VP_{\text{whale}} = \sqrt{1{,}000{,}000} \cdot 1.25 = 1{,}000 \cdot 1.25 = 1{,}250 \tag{13.46}$$

The capital ratio is 1,000:1 but the voting power ratio is only 23:1, a 97.7% compression of the capital advantage. This means small holders collectively wield meaningful governance influence.

13.6.5 xQORE Long-Term Savings

Individual users can lock QOR into xQORE to access PvP rebase yield without active management. The graduated exit penalty structure (50% at less than 30 days, 35% at 30 to 89 days, 15% at 90 to 179 days, 0% at 180 or more days) incentivises long-term commitment while providing liquidity under penalty for emergencies.

For a user locking 10,000 QOR for the full 180-day period, the expected return includes staking APY plus PvP rebase from early exits by other participants (Section 11.5). The rebasing mechanism transfers value from users exiting early to those remaining locked, aligning incentives for network security through staking participation. Users can compound their xQORE holdings by re-locking rebase yields, creating an exponential wealth accumulation curve for long-term holders. The smart contract enforces strict accounting to ensure that the sum of all xQORE balances plus redeemed penalties equals the original QOR locked, maintaining capital efficiency and preventing value creation outside the defined rebase structure.

Locked QOR holders gain governance rights within the xQORE module, allowing them to vote on parameter adjustments including penalty percentages, minimum lock periods, and rebase distribution mechanisms. This governance layer ensures that the savings mechanism evolves based on participant feedback while preserving the core incentive structure that aligns user interests with long-term network security.

13.6.6 NFT and Gaming

The RDK Gaming rollup profile provides optimised infrastructure for NFT marketplaces and blockchain gaming:

Gaming Rollup. Shared sequencer mode with optimistic settlement, targeting sub-second transactions for real-time gameplay. Game state is anchored to the main chain at configurable intervals, inheriting QoreChain's PQC security for in-game asset ownership.

NFT Rollup. Dedicated sequencer with ZK settlement for provably authentic digital art and collectibles. ML-DSA-87 signed provenance records ensure that NFT ownership history remains tamper-proof in the post-quantum era.

Cross-VM NFT Composability. NFTs minted on the EVM can be traded in CosmWasm marketplaces or used in SVM-based games through QoreChain's cross-VM messaging, providing unified liquidity across all three execution environments.

```
# Register as a light node (UX edition)
qorechaind tx lightnode register \
  --node-type ux \
  --from user-key \
  --output json

# Perform a cross-chain swap via One-Hop-Swap
qorechaind tx bridge swap \
  --from-chain ethereum \
  --from-token WETH \
  --to-chain solana \
  --to-token SOL \
  --amount 1000000000 \
  --route auto \
  --slippage-tolerance 0.01 \
  --from user-key \
  --output json

# Lock QOR into xQORE for savings
qorechaind tx xqore lock \
  --amount 10000000000uqor \
  --lock-duration 180 \
  --from user-key \
  --output json

# Create a gaming rollup
qorechaind tx rdk create-rollup \
  --name "QuestRealm Online" \
  --profile gaming \
  --settlement-mode optimistic \
  --vm-type evm \
  --sequencer-mode shared \
  --stake 10000000000uqor \
  --from game-studio-key \
  --output json
```

13.7 Cross-Chain DeFi and Liquidity

QoreChain's position as a quantum-safe hub connecting 25 chains creates unique opportunities for cross-chain decentralised finance applications.

13.7.1 Quantum-Safe DEX

A decentralised exchange deployed on QoreChain's EVM provides quantum-safe trading with natively PQC-protected order matching and settlement. Unlike DEXs on classical chains, QoreChain DEX orders and trades are signed with ML-DSA-87, preventing quantum-era front-running through signature forgery.

The automated market maker (AMM) constant-product invariant operates identically to classical DEXs:

$$x \cdot y = k, \qquad \Delta y = \frac{y \cdot \Delta x}{x + \Delta x} \tag{13.47}$$

The quantum-safe advantage is in the settlement layer: every swap proof is PQC-signed, and the 5-lane transaction prioritization (PQC lane at priority 100) ensures that PQC-signed transactions receive preferential ordering, reducing MEV exposure.

13.7.2 Cross-Chain Lending with PQC Collateral Proofs

Lending protocols on QoreChain can accept collateral from any connected chain, verified through PQC-signed bridge attestations. The collateral on the source chain is locked in a bridge escrow, and the PQC-signed attestation serves as proof of collateral on QoreChain:

$$\text{BorrowLimit}(u) = \sum_{c \in \mathcal{C}} \text{LTV}_c \cdot V_c(u) \cdot \mathbf{1}[\text{PQC\_Attestation}_c(u) \text{ valid}] \tag{13.48}$$

where $V_c(u)$ is the value of user u's collateral on chain c, and LTV_c is the chain-specific loan-to-value ratio reflecting bridge security risk. The PQC-signed attestation includes cryptographic proof that the underlying collateral remains locked on the source chain and has not been double-spent. Liquidation triggers occur when the borrower's collateral value falls below the borrow limit, with on-chain settlement executing through the bridge escrow to ensure atomic unwinding of multi-chain positions. The lending protocol supports cross-chain liquidation cascades where liquidation of collateral on one chain automatically updates borrowing capacity on QoreChain, providing real-time risk adjustment across the heterogeneous blockchain ecosystem. Interest rates are dynamically adjusted based on utilization ratios and chain-specific risk premiums, with governance able to pause new borrowing on high-risk chains while allowing existing borrowers to unwind positions. The protocol maintains a reserve pool funded by liquidation spreads and origination fees to cover rare scenarios where bridge failures prevent collateral redemption, ensuring protocol solvency across volatile cross-chain conditions.

13.7.3 Liquidity Depth Under One-Hop-Swap Topology

The One-Hop-Swap topology concentrates liquidity at the hub (QoreChain) rather than fragmenting it across bilateral bridges. The effective liquidity depth for a trading pair (A, B) where A and B are on different chains:

$$L_{\text{eff}}(A, B) = \min(L(A, \text{QOR}),\ L(\text{QOR}, B)) \tag{13.49}$$

This is strictly greater than the liquidity available through a direct bilateral bridge (which typically has lower TVL than the hub pools), and the single intermediate hop minimises slippage compared to multi-hop routes through multiple bridges. The liquidity concentration effect produces stronger network effects as the number of connected chains increases; with n connected chains, the hub supports $\binom{n}{2}$ trading pairs from only n liquidity pools, versus $\binom{n}{2}$ pools required by a complete bilateral bridge topology. Liquidity providers benefit from the concentration through higher fee capture per unit capital deployed; a $1 million pool on QoreChain can service trades from any of the 25 connected chains, yielding volume and fees that would require separate pools on each bilateral bridge endpoint. The topology also improves price discovery; arbitrageurs can efficiently balance prices across all connected chains by executing through the QoreChain hub, reducing fragmentation and benefiting all traders through tighter spreads and reduced slippage. QCAI route optimization algorithms dynamically direct order flow to pools with the deepest liquidity, further concentrating volume and improving execution for all participants.

13.7.4 Paychain Micropayments

The paychain layer (Chapter **??**) enables streaming micropayments with sub-500ms settlement for content consumption, API usage billing, and machine-to-machine payments. A paychain transaction costs a fraction of a main-chain transaction while inheriting the main chain's PQC security through periodic state anchoring:

$$C_{\text{paychain}} \approx \frac{C_{\text{mainchain}}}{N_{\text{batch}} \cdot (1 + \delta_{\text{anchor}})} \tag{13.50}$$

where N_{batch} is the number of paychain transactions per main-chain anchor and δ_{anchor} is the relative cost of the anchoring transaction.

```
# Add cross-chain liquidity to a DEX pool
qorechaind tx dex add-liquidity \
  --pool-id "QOR-ETH" \
  --amount-qor 100000000000uqor \
  --amount-eth 50000000000ibc/ETH \
  --slippage-tolerance 0.005 \
  --from lp-provider-key \
  --output json

# Register a streaming payment paychain
qorechaind tx multilayer register-paychain \
  --name "ContentStream Payments" \
  --target-block-time 500ms \
  --min-stake 100000000uqor \
  --from content-platform-key \
  --output json
```

13.8 Enterprise Private Chains and Compliance

Regulated enterprises require blockchain infrastructure that balances transparency with privacy, and decentralisation with compliance. QoreChain's multi-layer architecture provides purpose-built solutions for enterprise deployment.

13.8.1 RDK Enterprise Rollup

The RDK Enterprise rollup profile (Chapter **??**) provides:

- **Based settlement**: State anchored to QoreChain main chain for PQC-secured finality.
- **STARK proofs**: Cryptographic settlement proofs without trusted setup.
- **Dedicated sequencer**: Enterprise-controlled transaction ordering.
- **Native bridge**: Subsidised asset transfers between rollup and main chain.
- **1-second target latency**: Near-real-time transaction processing.

Enterprise deployments instantiate private or consortium rollups configured for specific use cases. A financial services consortium can deploy a rollup with permissioned validators owned by member banks, enabling them to execute thousands of transactions per second while settling batches to QoreChain mainnet every 60 seconds. The STARK proof system ensures cryptographic settlement without requiring participants to trust the rollup sequencer; the proof itself provides mathematical certainty that state transitions obey the rollup's rules. The dedicated sequencer can implement transaction ordering policies specific to the enterprise use case, such as fairness scheduling to prevent MEV or priority ordering based on transaction fees and execution timestamps. The native bridge subsidises transfers between the enterprise rollup and QoreChain mainnet, reducing friction for users who occasionally need to bridge assets to the main network or other rollups. The one-second target latency enables use cases requiring near-real-time settlement such as trading systems, payment processing, and settlement networks. Enterprise rollups inherit PQC security from the underlying QoreChain mainnet; settlement proofs are signed with ML-DSA-87, ensuring that rollup state remains quantum-safe even after deployment and operation.

13.8.2 Healthcare (HIPAA-Compliant Selective Disclosure)

Healthcare data management requires strict access control with auditability. The HCS layer enables HIPAA-compliant healthcare applications:

Electronic Health Records (EHR). Patient records are stored on an HCS instance operated by a healthcare consortium. Only PQC-signed state commitments are anchored on the main chain, ensuring that no protected health information (PHI) is exposed on the public ledger. Authorised providers access patient data through the HCS with PQC-authenticated selective disclosure:

$$\text{Disclose}(r, p) = \begin{cases} \text{Decrypt}_{\text{ML-KEM}}(r.\text{field}_i, sk_p) & \text{if Policy}(p, r.\text{field}_i) = \text{allow} \\ \perp & \text{otherwise} \end{cases} \tag{13.51}$$

where r is the health record, p is the requesting provider, and Policy encodes HIPAA minimum necessary access rules. Individual fields within a health record can be encrypted to different providers based on their role; a cardiologist can access cardiac history and imaging but not mental health records, while a psychiatrist has opposite permissions. The selective disclosure mechanism enforces fine-grained access control without exposing the entire record or revealing which fields exist. All access events are logged to an immutable audit trail anchored on QoreChain mainnet, providing proof of who accessed which records and when. Patient data itself never leaves the HCS instance; only the audit trail and selective disclosure events are recorded on the main chain, maintaining HIPAA compliance for storage and transmission. The HCS layer supports consent revocation; a patient can revoke a provider's access rights at any time, and the revocation is immediately propagated to all access control checks. Historical audit logs remain for regulatory record-keeping while future access is prevented.

13.8.3 Supply Chain Provenance

Manufacturing and logistics companies use QoreChain for end-to-end supply chain tracking with quantum-safe provenance guarantees:

Multi-Party Attestation. Each participant in the supply chain (manufacturer, shipper, customs, distributor, retailer) attests to their handling of goods using ML-DSA-87 signatures. The complete attestation chain provides an immutable, quantum-safe provenance record from origin to consumer.

Privacy-Utility Tradeoff. The HCS architecture enables supply chain participants to share provenance data selectively. Define the privacy level $\pi \in [0, 1]$ (0 = full disclosure, 1 = full privacy) and the utility $U(\pi)$ of the shared data:

$$U(\pi) = U_{\max} \cdot (1 - \pi)^{\alpha}, \qquad \alpha > 0 \tag{13.52}$$

The optimal privacy level balances competitive confidentiality with the transparency required by auditors and regulators:

$$\pi^* = \arg\max_{\pi} \left[\beta \cdot U(\pi) - \gamma \cdot \text{DisclosureRisk}(\pi)\right] \tag{13.53}$$

where β and γ are enterprise-specific weights on utility and risk.

13.8.4 Legal and Compliance

Timestamped Evidence. Legal proceedings increasingly accept blockchain timestamps as evidence. QoreChain's PQC-signed block commitments provide timestamps with 256-bit quantum security, ensuring that evidence anchored on-chain today remains admissible in future proceedings regardless of quantum computing advances.

KYC/AML Integration. Enterprise rollups can integrate KYC/AML verification through on-chain attestation: a compliance provider attests that an address has passed verification, and the rollup's access control layer checks this attestation before permitting transactions. The attestation itself contains no personal data, only a PQC-signed binary "verified/not verified" status with an expiry block height:

$$\text{KYC\_Attestation}(a) = \text{Sign}_{\text{ML-DSA-87}}(sk_{\text{provider}},\ a \| \text{status} \| h_{\text{expiry}} \| \text{jurisdiction}) \tag{13.54}$$

```
# Deploy an enterprise supply chain tracking HCS
qorechaind tx multilayer register-sidechain \
  --name "Pharma Supply Chain" \
  --layer-type hcs \
  --min-validators 3 \
  --pqc-commitment-only true \
  --from enterprise-key \
  --output json

# Submit a supply chain attestation
qorechaind tx attestation submit \
  --layer-id "pharma-supply-01" \
  --attestation-type "custody_transfer" \
  --data '{"product_id": "NDC-12345",
    "batch": "LOT-2026-042",
    "from": "manufacturer",
    "to": "distributor",
    "temperature_verified": true}' \
  --from manufacturer-key \
  --output json

# Issue a KYC attestation for an address
qorechaind tx attestation kyc-verify \
  --subject qor1user...addr \
  --status verified \
  --jurisdiction "CH" \
  --expiry-blocks 525600 \
  --from kyc-provider-key \
  --output json
```

Chapter 14

Development Roadmap

QoreChain's development spans nearly a decade of research, engineering, and architectural evolution. What began as infrastructure for a specific application domain grew into a general-purpose quantum-safe, AI-native Layer 1 blockchain through successive phases of experimentation, validation, and re-architecture. This chapter traces the full development trajectory from initial research through projected enterprise maturity, providing a transparent account of what has been achieved, what is in progress, and what lies ahead.

14.1 Development Philosophy

QoreChain's development follows three governing principles:

Security-First Engineering. Every architectural decision prioritises cryptographic security and protocol safety over feature velocity. Post-quantum cryptography is not an afterthought or migration target; it is a foundational design constraint applied from the protocol's earliest layers. This means accepting performance trade-offs and increased key material sizes to obtain quantum-safe guarantees. It also means continuous assessment of emerging threats; QoreChain incorporates formal verification of critical components and regularly updates threat models as cryptanalysis evolves.

Phased Delivery with Validation Gates. Each phase concludes with explicit validation criteria that must be satisfied before proceeding. No phase transition occurs without demonstrated evidence that the preceding phase's objectives have been met. This approach prioritises reliability over speed. Validation gates include performance benchmarking against targeted metrics, security audit completion, community testnet participation, and developer feedback on usability and integration. Each gate is transparent; failure requires re-planning the phase or modifying the approach rather than proceeding with unresolved issues.

Open-Core Transparency. QoreChain maintains a public open-source codebase for core protocol components, enabling community review, independent security auditing, and ecosystem development. Proprietary components (advanced AI models, bridge orchestration, performance optimisations) are developed separately and integrated through well-defined interfaces. This model provides transparency where it matters most (protocol security, validator code, bridge attestation) while allowing selective proprietary development in areas where competitive advantage or novel research warrants it. All interfaces between open and proprietary components are clearly documented and versioned to prevent vendor lock-in.

14.2 Phase 0: Research and Concept (Q4 2017 to 2019)

The project originated in Q4 2017 as research into blockchain infrastructure for the Digital Advertising and Services Token (DAST) platform. The founding team identified that no existing blockchain could meet the combined requirements of high-throughput transaction processing, cross-chain interoperability, and smart contract automation needed for advertising tracking and fraud detection.

14.2.1 Problem Identification

The digital advertising industry presented a compelling blockchain use case: programmatic advertising generates billions of micro-transactions daily, requires real-time fraud detection, and involves complex multi-party settlement across advertisers, publishers, ad networks, and verification services. Existing platforms fell short in multiple critical dimensions. The founding team identified that no single blockchain technology could simultaneously satisfy the conflicting requirements of the advertising ecosystem without fundamental architectural innovation.

- Ethereum's throughput (∼15 TPS in 2017) and gas fees made micro-transaction processing economically infeasible for advertising workloads. With impression tracking generating millions of micro-transactions daily, per-transaction costs of $0.50 to $5.00 made blockchain-based settlement uneconomical compared to traditional ad tech infrastructure. The economic case required throughput exceeding 5,000 TPS with fees under 0.1 cents per transaction.

- Alternative Layer 1 platforms lacked the cross-chain interoperability required to settle across fragmented advertising ecosystems. Digital advertising value flows across multiple blockchain ecosystems as advertisers, publishers, and ad networks operated on heterogeneous infrastructure. No existing bridge technology could reliably settle multi-chain transactions with the security guarantees required for financial transactions spanning independent consensus systems.

- No platform offered integrated AI capabilities for real-time fraud detection at the protocol level. The team recognised that algorithmic fraud detection, trained on historical advertising patterns, could identify suspicious transactions in real-time. Existing blockchain platforms treated AI as an external service rather than a protocol-level capability, creating latency and trust issues. The vision was to embed intelligence directly into consensus and transaction routing.

- Consensus mechanisms in existing platforms lacked the flexibility to balance security with participation incentives. Traditional Proof of Stake favoured capital accumulation, concentrating validator sets into fewer large holders. The advertising ecosystem required incentive structures rewarding long-term reliable operation and reputation, not just capital size.

This problem identification exercise established the foundational requirements driving Phase 1 development and the strategic vision guiding subsequent phases. Rather

than optimizing a single dimension, the team recognised that general-purpose infrastructure required simultaneous innovation across consensus, interoperability, economic incentives, and intelligent protocol operation.

14.2.2 Cosmos SDK Selection

After evaluating available blockchain frameworks, the team selected the Cosmos SDK (the version available in 2019) as the development foundation. The SDK's modular architecture, application-specific blockchain philosophy, and built-in IBC (Inter-Blockchain Communication) support provided the flexibility required to build custom consensus, execution, and interoperability layers.

The Cosmos SDK offered critical architectural advantages unavailable in monolithic blockchain frameworks. Its module-based design isolated business logic into composable, testable units without tight coupling to consensus or networking layers. The application-specific blockchain approach allowed developers to define custom state machines and governance rules rather than inheriting the constraints of general-purpose platforms. Built-in account abstraction, permission systems, and querying interfaces accelerated development of complex multi-layer systems. The standardised IBC protocol enabled cross-chain communication as a first-class protocol component rather than an afterthought, directly supporting the architectural vision of cross-chain settlement and interoperability.

The modularity proved crucial during Phase 3 when the quantum pivot required wholesale cryptographic re-architecture. Rather than rewriting core consensus logic or networking stacks, the team could replace the signature verification module and key derivation functions while preserving the application framework, validator set management, and transaction routing logic. This modularity provided the flexibility to adopt post-quantum algorithms without abandoning the substantial engineering investments in operational validation, bridge infrastructure, and governance mechanisms built during Phase 1.

This decision proved prescient: the Cosmos SDK's modular architecture later enabled the radical re-architecture required for post-quantum cryptography without discarding years of accumulated engineering. Alternative frameworks lacking this modularity would have required greenfield rewrites at significantly higher cost and delay.

14.2.3 Early Research Outputs

This phase produced foundational research in three areas that would define the project's long-term architecture:

Consensus Design. Initial exploration of hybrid consensus mechanisms combining stake-weighted selection with reputation and delegation metrics, laying the conceptual groundwork for what would become Combined Proof of Stake (CPoS).

Multi-Layer Processing. Theoretical work on separating transaction processing into distinct tiers (settlement, computation, high-frequency payments), recognising that a single execution layer could not simultaneously optimise for all workload types.

Cross-Chain Interoperability. Research into protocol-agnostic bridging that could operate across heterogeneous blockchain architectures, extending beyond the Cosmos IBC ecosystem to EVM-compatible chains and other Layer 1 networks.

14.3 Phase 1: DAST Development and Architecture (2019 to 2021)

With the Cosmos SDK as foundation, active development of the DAST platform began in 2019. This phase transformed the research concepts from Phase 0 into working software.

14.3.1 Core Architecture Implementation

The DAST platform implemented three architectural innovations that remain central to QoreChain today:

Combined Proof of Stake (CPoS). The first implementation of the hybrid consensus mechanism combining Reputation Proof of Stake (RPoS), Delegated Proof of Stake (DPoS), and traditional Proof of Stake (PoS). Validators were classified into pools based on delegation ratios and reputation scores, with block proposer selection weighted across all three pools. This design addressed the centralisation tendencies of pure stake-weighted selection.

Multi-Layer Transaction Processing. The main chain, sidechain, and paychain architecture was implemented as a working system. The main chain handled settlement and governance, sidechains provided application-specific execution environments, and paychains enabled high-frequency micropayments for advertising impression tracking.

DAST Bridge (6 Chains). The first cross-chain bridge connecting 6 blockchain networks was developed and deployed. The DAST Bridge established the engineering patterns for validator-attested cross-chain event verification, multi-signature confirmation thresholds, and chain-specific watcher services that would later evolve into the QoreChain Bridge (QCB) protocol. This early bridge implementation demonstrated the feasibility of protocol-agnostic bridging and provided critical operational experience with the challenges of cross-chain security, confirmation depth management, and bridge circuit breaker design.

14.3.2 Operational Validation

The DAST network was fully functional during this period, processing transactions across the multi-layer architecture and routing cross-chain transfers through the 6-chain bridge. This operational validation phase served as critical proof-of-concept for the technologies and architectural innovations proposed in Phase 0. Real-world network operation revealed both the feasibility of the multi-layer design and areas requiring refinement before broader deployment. The team validated consensus safety under realistic Byzantine fault injection, bridge operation under high throughput conditions, and the economic sustainability of multi-party settlement across chain boundaries.

This operational experience generated invaluable data on:

- Consensus behaviour under variable validator sets and network conditions.
- Bridge reliability across heterogeneous chain architectures with different finality models.
- Multi-layer routing efficiency and the tradeoffs between settlement security and transaction latency.

- Real-world smart contract execution patterns for advertising fraud detection and settlement automation.

14.3.3 Strategic Recognition

By 2021, operational experience with DAST revealed the fundamental limitations of application-specific blockchains. The engineering patterns developed for advertising settlement, fraud detection, and cross-chain payments were orthogonal to domain-specific logic. The combined proof-of-stake consensus mechanism, which dynamically balanced validator centralisation pressures across three separate pools, applied equally well to payment networks, supply chain tracking, or any system requiring Byzantine fault tolerance with decentralised validator participation. The multi-layer architecture that separated settlement concerns from computation from high-frequency operations addressed architectural tradeoffs inherent to any blockchain: no single execution layer could simultaneously optimise for settlement finality, smart contract expressivity, and throughput.

The cross-chain bridge infrastructure, developed to route digital advertising transactions across ethereum and cosmos-ecosystem chains, operated on protocol-agnostic principles. Event verification, validator attestation, multi-signature confirmation, and circuit breaker mechanisms did not depend on advertising-specific logic. This abstraction meant that the same bridge architecture could settle payments, atomic asset swaps, or any application requiring reliable value transfer across chain boundaries.

The team recognised that the core infrastructure being developed represented general-purpose blockchain technology applicable to any domain requiring high-throughput, cross-chain operations with strong cryptographic finality guarantees. This realisation motivated a strategic pivot away from domain-specific development toward building foundational infrastructure capable of serving diverse application ecosystems. The decision created a natural inflection point: rather than optimise further for advertising use cases, the team refocused on extending the platform's scope, strengthening its security properties, and preparing its architecture to survive emerging threats like quantum computing.

14.4 Phase 2: Foundation and Security Research (2021 to 2023)

This phase marked the project's evolution from an advertising-specific platform to general-purpose Layer 1 blockchain infrastructure, accompanied by deep research into security architecture and artificial intelligence integration.

14.4.1 First Genesis and Foundation Work

The first genesis configuration was established, formalising the network's initial state, validator set parameters, and module configuration. This genesis block represented the transition from ad-hoc development to formal protocol specification, documenting the precise state from which all subsequent consensus history would derive. The genesis specification included not only the token distribution and validator set but also the formal parameters governing all protocol operations: maximum block size, target block

time, transaction fee schedules, and the cryptographic constants underlying signature verification and state proofs. Creating this specification required translating empirical observations from Phase 1 into canonical protocol parameters that would govern validator behaviour for years to come.

Foundation work included:

- Formalisation of the CPoS consensus parameters (pool weights, reputation scoring coefficients, bonding curve formulas).
- Design of the governance module for on-chain parameter management.
- Specification of the token economics model, including emission schedules and fee distribution.
- Establishment of the validator onboarding process and delegation mechanics.

14.4.2 Security Architecture Research

Growing awareness of sophisticated attack vectors across the blockchain industry (bridge exploits exceeding $3.8 billion in losses since 2020) drove intensive security research:

Bridge Security. Analysis of the Ronin ($625M), Wormhole ($326M), Nomad ($190M), and Harmony Horizon ($100M) bridge exploits informed the design of multi-layer bridge verification, adaptive confirmation depths, and circuit breaker mechanisms.

Consensus Security. Research into validator collusion, long-range attacks, and stake centralisation led to the development of progressive slashing with temporal decay and the reputation-weighted proposer selection that would become core CPoS components.

14.4.3 AI Integration Research

The rapid advancement of machine learning capabilities presented an opportunity to integrate intelligence directly into blockchain protocol operations. During Phase 2, the team conducted comprehensive research into how AI could enhance security, efficiency, and user experience at the protocol level, recognising that post-quantum cryptography alone would not be sufficient to address the full spectrum of emerging threats to blockchain systems.

Transaction Routing. Research into reinforcement learning for dynamic transaction routing across the multi-layer architecture, optimising for throughput, latency, and fee efficiency simultaneously. The core insight was that the optimal routing policy depends on real-time conditions: transaction load, gas prices, latency, and security considerations all vary dynamically. Rather than fixed routing rules, an RL agent could learn optimal policies from observed outcomes, producing recommendations tailored to individual transaction characteristics and user preferences. This research informed the later design of the QCAI routing engine with support for fee, speed, security, and balanced optimisation modes.

Anomaly Detection. Exploration of statistical and machine learning methods for real-time detection of fraudulent transactions, bridge exploits, and consensus manipulation attempts. The team evaluated techniques including isolation forests, autoencoders, and temporal sequence models for detecting deviations from normal behaviour

patterns. This research demonstrated that AI-driven anomaly detection could identify attack patterns with false positive rates below 0.1% while catching 95

Validator Optimisation. Research into using ML models to dynamically adjust consensus parameters (block size, gas limits, pool weights) based on observed network conditions. The objective was to maintain safety guarantees while adapting to actual network conditions. This research led to the design of the RL agent in CPoS, which operates within explicit safety bounds (enforced by the circuit breaker) while learning policies that maximize finality latency under varying validator set compositions and network conditions.

14.4.4 Quantum Threat Awareness

During this period, quantum computing milestones from Google (Sycamore, 53 qubits, 2019), IBM (Eagle, 127 qubits, 2021), and IonQ (algorithmic qubits scaling) made the quantum threat to blockchain cryptography increasingly concrete. These milestones, combined with published cryptanalysis showing that ECDSA could be broken by a quantum computer with $\sim$ 2,330 logical qubits (compared to current systems with 100–500 physical qubits), crystallised the urgency of post-quantum migration.

The team began evaluating post-quantum cryptographic algorithms as potential replacements for the ECDSA signatures used throughout the protocol. The evaluation criteria included security level, performance overhead, key and signature sizes, and alignment with emerging standards. By 2022, NIST's third round of post-quantum cryptography standardisation had narrowed the candidate set, and the team began focusing on lattice-based schemes (Dilithium, Kyber), recognising their strong security arguments, superior performance characteristics, and likely standardisation.

A critical realisation from this research phase was that retrofitting PQC onto an existing blockchain architecture creates integration challenges: hybrid signature schemes introduce complexity, algorithm agility requires governance infrastructure, and the transition period during which both classical and PQC signatures coexist increases implementation risk. This recognition directly motivated the decision to undertake a full re-architecture in Phase 3, where PQC would not be an afterthought but a core design constraint from day one.

14.5 Phase 3: Quantum Pivot and Re-Architecture (2023 to 2024)

The accelerating quantum computing timeline transformed the project's strategic direction. Rather than retrofitting quantum resistance onto the existing architecture as an incremental upgrade, the team made the decision to completely re-architect the cryptographic foundation with post-quantum security as a core design constraint.

14.5.1 Post-Quantum Cryptographic Architecture

The re-architecture adopted three NIST post-quantum algorithms at their highest security levels:

- **ML-DSA-87** (Dilithium-5, FIPS 204): All digital signatures across consensus, transactions, bridge operations, and smart contracts.

- **ML-KEM-1024** (Kyber, FIPS 203): Key encapsulation for secure key exchange, bridge channel establishment, and custodial key rotation.
- **SHAKE-256**: Post-quantum resistant hash function for state proofs, Merkle trees, and address derivation.

These algorithms were selected based on security analysis of the underlying lattice problems (Module-LWE and Module-SIS), performance characteristics on target hardware, and alignment with the emerging NIST standardisation process. NIST's finalisation of FIPS 203, 204, and 205 in August 2024 validated these choices: the algorithms QoreChain had adopted were formally standardised for government and industry use.

14.5.2 Hybrid Migration Path

A hybrid classical-PQC protocol was designed to enable gradual migration from classical ECDSA to PQC-only operation. The design recognises that immediate PQC-only deployment would require ecosystem participants (users, exchanges, custodians, developers) to migrate infrastructure simultaneously, creating significant coordination challenges. The three-phase migration path is governance-controlled, allowing the network to transition at a pace determined by community readiness while maintaining backward compatibility during the transition period.

Phase 1: Classical (Genesis to Year 1). All transactions use classical Ed25519 signatures. PQC infrastructure is deployed and validated, but not yet required. This phase allows time for ecosystem tooling, wallets, exchanges, and libraries to integrate PQC support without breaking existing applications.

Phase 2: Hybrid (Year 1 to Year 3). Transactions may use either classical Ed25519 or PQC (ML-DSA-87) signatures. Validators and bridge attestations require PQC signatures, providing protection against quantum threats for consensus and cross-chain operations. Applications may transition to PQC at their own pace.

Phase 3: PQC-Only (Year 3 onwards). All transactions and consensus operations use PQC signatures. Classical signatures are no longer accepted. The transition to PQC-only is approved by governance (Tier 1 Constitutional proposal), ensuring community consensus before deprecating classical cryptography.

This staged approach provides protection against quantum threats where needed most (consensus, bridge, custody) while allowing gradual ecosystem migration. By Year 3, the quantum threat timeline and technological maturity will inform the decision to move to PQC-only, with governance retaining the option to extend the hybrid period if necessary.

14.5.3 AI-Native Design

In parallel with the quantum re-architecture, the AI integration research from Phase 2 was formalised into the QCAI (QoreChain AI) protocol layer. Rather than treating AI as an external service or optional feature, QCAI was designed as a native protocol component integral to security, routing, and economic functions. The design recognises that modern blockchain security requires real-time anomaly detection at scales (millions of transactions per day) where statistical rule-based systems become impractical.

QCAI is architected as a three-tier system for different computational and latency requirements:

QCAI Fast. Extremely low-latency inference for time-critical operations like route selection and transaction classification. Uses lightweight models (typically 10–50 MB) deployed directly in validator nodes.

QCAI Balanced. Moderate latency inference for fraud scoring and anomaly detection. Uses medium-scale models (100–500 MB) running on dedicated off-path QCAI services alongside validators.

QCAI Advanced. High-accuracy inference for complex pattern recognition and governance parameter optimisation. Uses large models (1–5 GB) running on dedicated GPU-equipped QCAI nodes with batch processing.

The off-path service architecture was deliberately chosen to enable AI inference without burdening the consensus-critical path. Validators perform their core consensus duties unaffected by QCAI computation; QCAI services run in parallel, with their outputs providing recommendations and context for protocol decisions rather than hard requirements. This architecture ensures that consensus remains safe and responsive even if QCAI services degrade or are unavailable.

14.5.4 Rebranding

The project was rebranded to reflect its evolved scope and advanced technology direction. The original DAST (Digital Advertising and Services Token) branding had always been organisational shorthand rather than the project's true technological identity. By 2023, it was clear that the infrastructure being built represented something far broader: a quantum-safe, AI-native settlement layer applicable to any domain requiring high-throughput, cross-chain operations with cryptographic guarantees extending decades into the future.

The rebranding process involved several strategic considerations. The advertising use case, while valuable, had become a narrow application of the underlying infrastructure. The platform's core capabilities around post-quantum cryptography, multi-chain settlement, and AI-integrated protocols were orthogonal to advertising-specific concerns. Continuing under the DAST branding created cognitive barriers with potential users in other industries who might not recognise their own use cases in advertising-focused messaging. The technical advances achieved during the quantum pivot and re-architecture represented a material scope change that warranted new branding to signal to the broader blockchain community that the platform had expanded well beyond incremental improvements to existing systems.

The name "QoreChain" was chosen to convey the project's position as core infrastructure ("Qore") for the post-quantum blockchain ecosystem ("Chain"). The terminology emphasises both the technical foundation and the scope: post-quantum cryptography as foundational rather than optional, and enabling infrastructure for the entire crypto ecosystem rather than a single application. The visual identity and technical documentation were updated to emphasise the unique architectural properties: quantum-safe signatures; native AI integration; multi-layer processing enabling simultaneous settlement, computation, and high-frequency operations; and comprehensive cross-chain support enabling seamless interoperability across heterogeneous blockchain networks.

The rebranding communicated to the broader blockchain ecosystem that QoreChain

was no longer application-specific infrastructure but rather a general-purpose Layer 1 platform whose architecture was specifically designed for the post-2030 environment where quantum computing and advanced AI capabilities would both be fully present. This repositioning significantly expanded the addressable market and attracted attention from institutional users and enterprise deployments requiring long-term cryptographic security guarantees.

14.6 Phase 4: Platform Development (2024 to 2025)

With the re-architecture complete, Phase 4 focused on implementing the full QoreChain platform across all major subsystems.

14.6.1 Core Module Development

Custom modules were developed covering the full spectrum of QoreChain's capabilities. This development work represented the consolidation of research and validation work from previous phases into production-grade implementations suitable for long-term network operation. Module development required deep integration with the Cosmos SDK's Application Blockchain Interface (ABCI), token economics simulation and validation, formal security review of cryptographic implementations, and comprehensive testing under edge cases and Byzantine fault conditions.

Each module required specification of module state and transaction types, implementation in Go language integrating with ABCI, and comprehensive testing including unit tests, integration tests, property-based testing, and fuzzing against malformed inputs.

- **Cryptographic layer**: `x/pqc` (ML-DSA-87 and ML-KEM-1024 integration, hybrid signatures, algorithm agility framework).
- **AI layer**: `x/ai` (QCAI integration, tiered inference, anomaly detection).
- **Consensus layer**: `x/qca` (CPoS pool classification, bonding curve, progressive slashing, QDRW governance), `x/rlconsensus` (on-chain RL agent, circuit breaker).
- **Economic layer**: `x/burn` (10-channel burn engine), `x/xqore` (governance-boosted staking), `x/inflation` (epoch-based emission decay).
- **Interoperability layer**: `x/bridge` (QCB protocol, multi-chain support), `x/multilayer` (sidechains, paychains, HCS), `x/rdk` (Rollup Development Kit).
- **Infrastructure layer**: `x/lightnode` (SX/UX editions, 3% fee share), `x/license` (27 feature IDs for bridge and validator operations), `x/abstractaccount` (spending rules, session keys).

14.6.2 Triple-VM Integration

The triple-VM execution environment (EVM + CosmWasm + SVM) was implemented with cross-VM messaging, enabling atomic transactions spanning all three virtual machines. Each VM serves distinct use cases and developer communities:

EVM (Ethereum Virtual Machine). Deployed for Ethereum compatibility, enabling direct porting of Solidity dApps. The EVM module handles gas accounting, state storage, and contract deployment identical to Ethereum's semantics, allowing developers and users familiar with Ethereum tooling to operate without learning new interfaces.

CosmWasm. Deployed for secure, auditable smart contracts using Rust-based WASM modules. CosmWasm's capability-based security model and formal verification support make it the preferred environment for mission-critical financial and custody applications where safety properties must be formally verified.

SVM (Solana Virtual Machine). Integrated to enable parallel transaction execution and high-throughput program operations. The SVM implementation includes a Berkeley Packet Filter (BPF) bytecode interpreter, Solana's account-based state model, and Solana's parallel runtime scheduler, enabling native execution of Solana programs on QoreChain.

Cross-VM messaging enables atomic transactions that span multiple VMs. For example, an EVM user could lock assets in an EVM smart contract, with a single atomic transaction that simultaneously executes a CosmWasm-based swap and mints the result in the SVM. The implementation uses a transaction wrapper that decomposes the cross-VM operation into sequential sub-transactions, each validated in its respective VM, with an all-or-nothing semantics enforced at the protocol level.

The SVM integration included a complete BPF execution engine with Solana-compatible JSON-RPC endpoints, comprehensive test coverage against known Solana programs, and performance benchmarking demonstrating sub-millisecond per-transaction overhead for BPF execution relative to native Cosmos modules.

14.6.3 Bridge Expansion

The original 6-chain DAST Bridge was significantly expanded, growing from a single-purpose bridge serving the advertising ecosystem to a comprehensive cross-chain interoperability layer serving all major blockchain ecosystems. The bridge infrastructure grew from the initial 6-chain proof-of-concept to 25 total chain connections:

IBC Channels (8 Cosmos Chains). Standard Inter-Blockchain Communication (IBC) channels connecting QoreChain to the Cosmos ecosystem: Cosmos Hub (primary economic hub), Osmosis (major DEX), Noble (native USDC issuer), Celestia (data availability layer), Stride (liquid staking protocol), Akash (decentralised compute), Babylon (BTC restaking), and an internal loopback channel for testing.

QCB Bridge Endpoints (17 Non-Cosmos Chains). The QoreChain Bridge (QCB) protocol, described in Chapter 9, provides bridges to major non-IBC chains spanning multiple architectures: EVM chains (Ethereum, BSC, Polygon, Arbitrum, Optimism, Base, Avalanche), Solana, Solana-fork (TON, NEAR), UTXO chains (Bitcoin), Move-based (Aptos, Sui), Cardano, Polkadot, Tezos, and TRON.

This 25-chain connectivity realises the One-Hop-Swap principle: any pair of connected chains can execute a direct transfer through QoreChain with exactly one intermediate hop, avoiding the multi-hop routes and compound risks of general cross-chain

networks. The routing topology is formally a star graph with QoreChain at the center, providing superior cost and reliability characteristics relative to multi-tier bridging networks.

The QCAI routing engine was integrated to provide intelligent selection between IBC and QCB routes for Cosmos chains reachable via both protocols. This dual-protocol coverage ensures redundancy: if IBC relayer latency increases or QCB validators fall below quorum, the alternative protocol remains available, improving overall system reliability. Bridge validation follows the PQC-signed attestation model; every bridge validator's attestation is signed with ML-DSA-87, ensuring that bridge consensus is quantum-safe even when validators operate using classical infrastructure. The bridge expansion required hardening of the bridge validator set with geographic and organizational diversity; validators are operated by independent parties rather than a single organization, reducing systemic risk from compromise of any single validator node. The two-month bridge expansion period included stress testing with adversarial transaction patterns, simulating conditions of extreme market volatility and coordinated attack vectors. The testnet bridge validator set is designed to handle multiples of expected steady-state load while maintaining sub-second settlement latency and quorum availability targeting institutional service-level expectations.

14.6.4 Swiss Foundation Establishment

In November 2025, the QoreChain Association was incorporated in Rolle, Switzerland (CHE-484.963.998), establishing the regulatory framework and jurisdictional stability required for institutional adoption and enterprise partnerships. This formal entity provides the legal structure and regulatory compliance necessary for partnerships with regulated financial institutions and government bodies.

The Swiss foundation structure was selected based on several factors: Switzerland's leading position in blockchain regulation through the pioneering Financial Services Regulation Act (FinSA) and Anti-Money Laundering Act (AML); the foundation's non-profit structure, which aligns with QoreChain's open-source, community-driven governance model; and Switzerland's stable political and financial system, providing long-term institutional credibility spanning centuries of financial stewardship.

The QoreChain Association provides:

- **Regulatory Clarity**: Registration under Swiss blockchain legislation (DLT Act), providing clear legal status for token issuance, node operation, and foundation activities. The foundation maintains AML/KYC compliance infrastructure for regulated entity partnerships.

- **Institutional Credibility**: A legally recognised organisation suitable for partnerships with banks, exchanges, central banks, and government agencies. The foundation can execute commercial contracts, intellectual property licensing, and regulatory cooperation agreements that require a formal legal entity.

- **Token Issuance Framework**: Compliance with Swiss financial regulations for token generation and distribution. The foundation maintains records of all token allocations, lockup periods, and vesting schedules, satisfying regulatory transparency requirements.

- **Governance Separation**: Foundation governance (board of directors, member elections) is separate from protocol governance (duration-weighted voting, with the optional QDRW extension), ensuring long-term stewardship of core infrastructure independent of token holder sentiment. The foundation operates with fiduciary duties to the ecosystem, preventing capture by individual token holders.
- **Strategic Asset Management**: The foundation holds key intellectual property, domain names, and trademarks, protecting them from individual control and ensuring continuity regardless of personnel changes.

14.6.5 Single-Node Validation

All modules were compiled, wired into the application framework, and validated through comprehensive unit testing. The single-node testnet confirmed correct genesis initialisation, module interaction, transaction processing, and state management across all subsystems.

The single-node validation phase focused on four critical areas. **Consensus Safety**: Verified that block proposal, voting, and finalisation follow the CPoS rules with reputation scoring, bonding curve rewards, and progressive slashing functioning correctly. Specifically, testing confirmed that block time remained stable at the target 2-second interval, that the RL agent's policy learning produced stable consensus parameters without oscillation, and that slashing calculations correctly identified and penalised misbehaving validators. **Module Interactions**: Confirmed that fee distribution across ten channels occurred with correct accounting, that burn channels properly removed tokens from circulation, that emission schedules scaled correctly through epoch transitions, and that staking and governance modules interacted without state inconsistencies or double-spending vulnerabilities. **Cross-VM Execution**: Validated that EVM, CosmWasm, and SVM modules could execute within a single block with proper state isolation, that gas accounting across VMs provided consistent pricing regardless of execution environment, and that cross-VM messaging semantics matched the atomic transaction guarantees. **Bridge Infrastructure**: Verified that bridge validators could propose and verify attestations, that QCAI anomaly detection operated correctly without false positives on normal transaction patterns, that the circuit breaker triggered appropriately under simulated attack conditions, and that circuit breaker resets followed the designed temporal decay schedule.

Performance profiling on single-node revealed that the PQC signature overhead, specifically ML-DSA-87 verification at approximately 330 microseconds per signature, was acceptable for the designed throughput targets of 5,000+ TPS when accounting for the parallel verification opportunities provided by the multi-node consensus implementation. The triple-VM execution environment successfully processed mixed EVM, CosmWasm, and SVM transactions with cross-VM messaging overhead measured at less than 2

14.7 Phase 5: Testnet Validation (Q1 to Q3 2026)

Phase 5 transitions from single-node development to multi-node network validation, the critical step between code completeness and production readiness.

14.7.1 Multi-Node Testnet Deployment

The primary objective of Phase 5 is deploying and operating a multi-node testnet with geographically distributed validators. This phase represents the transition from controlled single-node development to production-grade network validation with realistic network conditions, latency, and Byzantine fault scenarios. The multi-node testnet deployment validates that consensus safety and liveness properties proven theoretically actually hold under realistic network topologies and failure modes.

Key validation targets:

- **Consensus liveness**: Continuous block production across a validator set of 10+ nodes with simulated failures and network partitions.
- **Performance benchmarking**: Measurement of actual TPS, block time, finality latency, and PQC signature overhead under realistic transaction loads.
- **Cross-VM execution**: End-to-end validation of EVM, CosmWasm, and SVM transaction processing and cross-VM messaging in a multi-node environment.
- **Bridge testing**: IBC channel operation and QCB bridge endpoint verification with live testnet connections to partner chain testnets.

14.7.2 Light Node Connectivity

SX and UX light node editions are deployed alongside the multi-node testnet, testing their functionality under realistic network conditions. Light node testing validates critical infrastructure components including registration procedures where new nodes authenticate and join the active network; heartbeat mechanisms where nodes maintain liveness proofs to evidence continuous participation; and reward distribution calculations where the protocol attributes the 3

The SX (Standard) edition, designed for typical infrastructure deployments, validates performance characteristics including block header synchronisation time, state proof generation latency, and query response rates. Testing measures the computational overhead of cryptographic state proof verification (specifically, the additional cost of verifying ML-DSA-87 signatures embedded in state proofs rather than classical Ed25519 signatures) and confirms that PQC overhead does not degrade user experience.

The UX (Ultra-light) edition, targeting mobile devices and IoT platforms, undergoes stress testing on constrained hardware including memory budgets, network bandwidth requirements, and power consumption profiles. Test scenarios include temporary disconnections followed by synchronisation, interaction with degraded network conditions (high latency, packet loss), and operation across multiple different geographies to validate performance stability across international deployments.

Fee share distribution testing validates that the accounting mechanisms correctly attribute 3

14.7.3 Security Audits

Third-party security audits are conducted across three domains:

Cryptographic Audit. Independent review of ML-DSA-87 and ML-KEM-1024 implementations, hybrid signature scheme, and PQC state proof construction.

Consensus Audit. Review of CPoS implementation, RL agent safety constraints, circuit breaker design, and progressive slashing logic.

Bridge Audit. Security review of QCB protocol, bridge circuit breakers, attestation verification, and cross-chain message authentication.

14.7.4 Community Testnet

Following internal validation, the testnet is opened to community validators. The community testnet phase serves multiple critical purposes beyond technical validation. Community participation provides real-world feedback on operator documentation, tooling, and support channels, revealing gaps not evident from internal testing. External validators operating diverse infrastructure uncover integration challenges with specific cloud providers, networking stacks, and monitoring solutions. Community participation establishes the foundation for mainnet validator recruitment, enabling operators to demonstrate reliable performance on testnet and build reputation prior to main-chain launch.

The community testnet phase serves multiple purposes:

- Stress testing under adversarial conditions with external participants.
- Validator onboarding program: documentation, tooling, and support channels.
- QoreChain Studio beta release for developer testing and feedback.
- Governance module activation with community proposals on testnet parameters.

14.7.5 Validation Gate

The validation gate establishes objective criteria determining whether the network has demonstrated sufficient operational maturity and security assurance to proceed to mainnet launch. These criteria were defined during Phase 4 design and are evaluated against real testnet data collected during Phase 5 community participation. The validation gate prevents premature mainnet launch while maintaining schedule predictability.

Phase 5 concludes only when the following criteria are satisfied:

1. Multi-node testnet operates continuously for 30+ days without consensus failure.
2. Benchmarked TPS meets or exceeds the designed target under realistic load.
3. All three security audits complete with no critical or high-severity findings unresolved.
4. Light node network operates with 50+ nodes maintaining >80% uptime.
5. Bridge endpoints demonstrate successful cross-chain transfers on at least 5 partner chain testnets.

14.8 Phase 6: Mainnet Launch (Q2 2026)

Mainnet launch is contingent on successful completion of all Phase 5 validation gates. The launch sequence proceeds through a controlled activation schedule.

14.8.1 Genesis Block

The mainnet genesis block represents the culmination of years of research, development, validation, and community participation. Genesis initialisation is the irreversible moment at which the protocol transitions from testnet to production operation, where all decisions become immutable historical facts and every subsequent block depends on the correctness of genesis parameters. Genesis block creation requires exceptional care: the initial validator set must be sufficiently diverse to ensure no single entity controls consensus; token distribution must reflect fair allocation across contributors, community members, and strategic partners; governance parameters must be conservative enough to prevent reckless protocol changes while remaining flexible enough for the community to adapt to unforeseen circumstances.

The genesis state includes:

- Initial validator set selected from community testnet participants who meet minimum stake, uptime, and reputation requirements.
- Token distribution according to the approved allocation schedule.
- All governance parameters set to their default values with safety bounds active.
- PQC algorithm registry initialised with ML-DSA-87, ML-KEM-1024, and SLH-DSA.
- Bridge endpoints in staged activation mode (enabled per-chain as bridge validators come online).

14.8.2 Token Generation Event

The TGE marks the activation of the QOR token economy on mainnet, transitioning from a testnet environment with abstract tokens to production-grade economic mechanics. At genesis, the protocol initialises the complete token distribution according to the approved allocation schedule: community allocation recipients receive their initial token batches with appropriate vesting schedules; foundation treasury reserves are locked according to governance-approved multi-signature controls; and initial validator pools receive their designated allocations to ensure sufficient stake for consensus liveness.

Epoch-based emission begins at genesis according to the designed inflation schedule, specifically starting with the Year 1 emission rate of 17.5

The 10-channel burn engine activates simultaneously with emission, creating continuous token velocity dynamics. Transaction fees are distributed across ten burn channels according to their designated functions: settlement channel (35

The simultaneous activation of emission and burn creates the token supply equilibrium that characterises the QOR long-term economics. The interplay between these systems is designed such that the burn rate approximately balances the emission rate

by Year 3, after which the protocol transitions to net-deflationary operation where accumulated burn exceeds new emission.

14.8.3 Staged Bridge Activation

Bridge endpoints are activated sequentially rather than simultaneously, allowing operational verification of each chain connection before moving to the next. This staged approach reduces the risk of simultaneous bridge failures and provides time to diagnose and resolve operational issues with specific chain integrations.

1. **IBC Channels (Week 1).** Cosmos-ecosystem IBC channels (Cosmos Hub, Osmosis, and partner Cosmos chains) are activated first, as IBC has the most established operational history and the lowest risk of integration issues. Successful IBC activation provides proof that the hybrid classical/PQC attestation system works correctly.

2. **High-Priority QCB Endpoints (Weeks 2–4).** QCB endpoints for the highest-liquidity and most widely integrated chains (Ethereum, Solana, BSC) are activated, enabling primary use cases for institutional settlement and cross-chain asset transfers. These chains have the most sophisticated monitoring infrastructure and active security communities.

3. **Secondary QCB Endpoints (Weeks 5–12).** Remaining QCB endpoints are activated over the following weeks as bridge validators obtain licenses and demonstrate operational readiness. The staggered schedule allows the bridge team to focus security attention on one or two chains at a time, conducting thorough testing before moving to the next.

The staged activation includes explicit success criteria for each phase: successful token transfer in both directions within target latency (to be specified post-launch), circuit breaker functionality under load, and quorum achievement with licensed validators. If issues are discovered, an endpoint can be paused and re-tested without affecting already-activated chains.

14.8.4 QCAI and Governance Activation

QCAI services are activated in shadow mode initially (observe and log, no autonomous actions), transitioning to conservative mode after a governance proposal approving autonomous operation. This phased activation approach balances the security benefits of AI-driven anomaly detection against the risk of unforeseen AI model failures or unexpected interactions with real network conditions.

Shadow Mode (Weeks 1–4). QCAI models are deployed and operating on real network data, but all scores and anomalies are logged without blocking any transactions. During this period, QCAI maintainers monitor false positive rates, verify model behaviour against known threat patterns, and calibrate thresholds for the next phase. This provides valuable operational data without risking transaction blocking from poorly-calibrated models.

Conservative Mode (Weeks 5–12). After governance approval, QCAI transitions to conservative mode: only the highest-confidence anomaly detections trigger

transaction flagging or bridge rate limiting. The system is configured to minimise false positives, allowing the community to gain confidence in AI-driven security without risking legitimate transaction blocking.

Full Deployment (Month 4+). After demonstrated stable operation and community confidence, QCAI moves to full deployment with all three verification layers active for bridge transactions and full routing optimisation across all cross-chain operations.

The governance module is active from genesis, enabling the community to manage the activation schedule for remaining features. This represents a core design principle: QoreChain's infrastructure features are activated through transparent governance proposals rather than unilateral deployment decisions, preserving community control over the protocol's evolution.

14.8.5 Ecosystem Launch

Concurrent with mainnet launch, a comprehensive developer and user ecosystem is activated to enable immediate community participation and application development. The ecosystem launch represents the transition from protocol infrastructure to living, evolving platform. User-facing tools, documentation, and support channels must be production-grade from day one, as early developer and user experiences shape perceptions of platform maturity and usability. Ecosystem launch requires coordinated release of multiple components: developer platform, block explorer, light node network, documentation, SDKs, and community support infrastructure.

The ecosystem launch includes:

- **QoreChain Studio Public Release**: The complete developer platform with QCAI contract generation (automatic contract generation from natural language specifications), formal verification capabilities, integrated testing environment, and deployment pipelines for all three VMs (EVM, CosmWasm, SVM). Studio is released with comprehensive documentation and example templates for common patterns (token contracts, DEX, NFT collections, governance).

- **Block Explorer and Dashboard**: A full-featured block explorer at dashboard.qorechain.io providing real-time transaction tracking, validator performance metrics, QCAI anomaly detection statistics, bridge transfer tracking, and governance proposal tracking. The dashboard is released with REST and GraphQL APIs for third-party integrations.

- **Light Node Network Launch**: SX and UX light node editions are made available for deployment, with automated installer scripts for major operating systems and cloud platforms. A public registry of community-operated light nodes enables users to discover and connect to available nodes. Initial seed funding and infrastructure grants are provided through the treasury for early node operators.

- **Developer Documentation and SDKs**: Comprehensive documentation covering the protocol design, API specifications, and development best practices for all three VMs. Official SDKs for JavaScript/TypeScript, Rust, Go, and Python are released, enabling developers to build applications without deep protocol knowledge.

- **Community Onboarding**: Official validator documentation, testnet-to-mainnet migration guides, and community support channels (Discord, Telegram, forums) are activated, with dedicated community managers answering questions and helping onboard new validators.

14.9 Phase 7: Ecosystem Expansion (2027)

With mainnet operational and validated, Phase 7 focuses on building the application ecosystem and expanding the network's utility.

14.9.1 DeFi Protocol Deployment

Core DeFi primitives are planned for deployment on QoreChain's EVM and CosmWasm environments, positioning QoreChain as a sophisticated financial infrastructure platform. Phase 7 DeFi deployments are intended to demonstrate that QoreChain's architecture supports complex financial protocols with the security guarantees and performance characteristics required for institutional users. These deployments are designed to validate the practical utility of quantum-safe signatures for financial transactions, showing that PQC overhead does not impair user experience. DeFi protocol success is expected to establish QoreChain as a destination for builders seeking to create sophisticated financial applications with confidence in long-term cryptographic security.

Core DeFi primitives planned for deployment on QoreChain's EVM and CosmWasm environments include:

- **Quantum-Safe Decentralised Exchange (DEX)**: A full-featured DEX deployed on the EVM with concentrated liquidity pools (Uniswap V4 style), automated market maker mechanics, and swap routing optimised by QCAI across multiple pools and connected chains via One-Hop-Swap. All order matching and settlement is PQC-signed, ensuring long-term cryptographic validity of historical trades.

- **Cross-Chain Lending and Borrowing**: Protocols deployed on CosmWasm enabling users to deposit collateral on one chain and borrow stablecoins on another, with collateral proofs anchored on-chain and signed with ML-DSA-87. Interest rates are dynamically adjusted based on utilisation via an independent rate oracle service.

- **Stablecoin Platforms**: Multiple stablecoin designs leveraging QoreChain's settlement finality and security guarantees. These include over-collateralised stablecoins (e.g., xUSD backed by staked QOR), redemption-based stablecoins (algorithmic supply adjustment based on redemption pressure), and bridge-wrapped stablecoins representing assets locked on connected chains.

- **Yield Aggregation**: Protocols that automatically route user capital across QoreChain and connected chains to optimal yield sources. The One-Hop-Swap architecture enables a single transaction to deposit on Ethereum, swap on Osmosis, and farm yields on Solana, all atomically.

14.9.2 QoreChain Studio General Availability

QoreChain Studio is scheduled to reach general availability with production-grade features enabling enterprises and developers to deploy sophisticated applications across the QoreChain ecosystem. Key capabilities include full support for all 17 QCB-connected chain types, enabling developers to write once and deploy to any supported blockchain; QCAI Advanced tier contract generation producing formally-verified smart contracts from high-level specifications; and integrated deployment pipelines for EVM, CosmWasm, and SVM targets with automated gas optimisation.

The GA release includes: advanced debugging tools with step-through execution and state inspection for all three VMs; automated security scanning detecting common vulnerabilities and suggesting fixes; integration with popular auditing firms for coordinated security reviews; and a contract marketplace enabling developers to discover, audit, and integrate open-source components. Studio also includes an integrated simulator enabling developers to test smart contracts against realistic network conditions before deployment, including simulated failure modes (validator slashing, bridge failures) to validate application resilience.

Studio GA also introduces the QCAI Contract Generation Advanced tier, a generative model trained on thousands of formal verification proofs and security-audited contracts. Developers can specify contract behaviour in natural language ("A stablecoin that adjusts supply based on price moving outside a 10

14.9.3 Enterprise Pilot Programs

Structured pilot programs with institutional partners across target verticals are planned to provide real-world validation of use cases and to generate concrete evidence of QoreChain's value proposition to potential enterprise customers. Enterprise pilots are intended to serve dual purposes: validating that QoreChain's architecture can deliver value in production environments with real financial stakes, and generating case studies and testimonials that establish proof of institutional adoption. Pilots will require close collaboration with enterprise technical and business teams, with dedicated QoreChain engineers embedded in partner deployments.

The pilot programs focus on vertical-specific use cases where blockchain's properties (immutability, transparency, multi-party settlement) provide clear advantages over traditional infrastructure:

- **Financial Services**: Quantum-safe settlement infrastructure pilots with global systemically important banks (G-SIBs) and international clearinghouses. Pilots demonstrate sub-second settlement finality for high-value transfers, PQC-signed settlement proofs suitable for regulatory filing, and integration with legacy banking infrastructure via bridge connections.
- **Healthcare**: HIPAA-compliant HCS (Hidden Computation Sidechain) deployments for electronic health record (EHR) management with major health systems. Pilots validate selective disclosure mechanisms enabling providers to access patient data without exposing PHI on public blockchains, and demonstrate interoperability enabling patients to control data access across providers.
- **Supply Chain**: Multi-party provenance tracking pilots with major manufacturing and logistics companies spanning automotive, pharmaceutical, and food

supply chains. Pilots demonstrate quantum-safe attestation of product custody and handling, enabling consumers to verify product authenticity and regulators to audit supply chains post-recall.

- **Government**: CBDC infrastructure evaluation with central bank research departments and payments authorities. Pilots explore wholesale and retail CBDC issuance on QoreChain infrastructure, demonstrating programmable currency, programmable payments, and atomic settlement of cross-border transactions.

Each pilot program includes dedicated technical support, customised deployment architecture, and documented lessons learned that inform product development priorities and marketing messaging for broader enterprise adoption.

14.9.4 IoT and Mobile

The lightweight quantum-safe wallet is planned for release as a production-grade SDK and reference implementation supporting constrained device deployments. The wallet is designed for IoT devices and mobile platforms with stringent computational, memory, and power constraints, enabling secure quantum-safe transaction signing without requiring full blockchain node infrastructure.

The wallet architecture separates concerns into distinct layers: a cryptographic layer implementing ML-DSA-87 signature generation and ML-KEM-1024 key encapsulation in fixed-size buffers optimised for embedded devices; a transaction layer that builds signed transactions without requiring state synchronisation; and a connectivity layer that uses standard HTTP and WebSocket protocols to communicate with light nodes or full nodes without requiring custom network stacks.

Device identity attestation enables IoT networks to establish cryptographic identity for each device, allowing devices to prove their identity and state to external parties using post-quantum signatures. The attestation format compresses identity proofs into compact payloads suitable for transmission over bandwidth-constrained networks. For example, a sensor device can generate an attestation proving it measured a specific value at a specific timestamp, signed with its ML-DSA-87 key, in approximately 3-5 KB of data including all cryptographic material.

Micro-transaction signing enables IoT devices to conduct direct value transfers or payments without requiring trusted intermediaries. The wallet implements session key mechanisms allowing devices to sign multiple transactions under a single authorisation, reducing the number of expensive signature operations required for high-frequency devices. For device deployments requiring hundreds of daily transactions, session keys can reduce the signature operation count by up to 90

The wallet SDK is provided in multiple implementations optimised for different platforms: a Rust implementation for embedded systems and edge compute; a JavaScript/TypeScript implementation for mobile platforms and web-based applications; and a C implementation for legacy and resource-constrained hardware where even Rust's binary size is prohibitive.

14.9.5 RDK Rollup Launch

The Rollup Development Kit (RDK) is scheduled to become generally available as a comprehensive platform for launching sovereign rollups secured by QoreChain. The

RDK is planned to ship with four preset profiles covering the most common rollup use cases, significantly reducing deployment complexity and time-to-market:

- **DeFi Profile**: Pre-configured for DEX, lending, derivatives, and other financial protocols. Includes integrated price oracle infrastructure, automated liquidation mechanisms, and cross-rollup liquidity routing.
- **Gaming Profile**: Optimised for game state management with low-latency block times (target 500ms), high throughput for concurrent player actions, and integrated NFT minting for in-game assets.
- **NFT Profile**: Specialised for NFT marketplaces and digital ownership platforms, with optimised metadata storage, creator royalty enforcement, and integrated secondary market mechanisms.
- **Enterprise Profile**: Built for private/permissioned deployments with access controls, data privacy mechanisms, and regulatory compliance features (KYC/AML integration, audit trails).

The RDK launch will be accompanied by: detailed deployment documentation with cost estimates and timeline projections; a rollup operator onboarding program providing technical support; approved sequencer and prover implementations with security audits; and a community funding program providing grants to teams launching rollups solving compelling user problems.

First-wave rollup deployments from ecosystem partners are expected to demonstrate the RDK's viability and generate network effects as applications deploy across multiple rollups, benefiting from shared liquidity pools and unified wallet interfaces. These early deployments are intended to establish rollup ecosystems as core QoreChain infrastructure components, expanding the network's total addressable market beyond the main chain.

14.9.6 Developer Ecosystem

A comprehensive developer ecosystem is planned to attract builders and accelerate application development across the network. Developer success is critical to QoreChain's long-term adoption: developers create applications that generate network utility, attract users to the platform, and demonstrate practical value propositions. The developer ecosystem investment will provide multiple forms of support: direct financial grants enabling teams to focus on application development rather than fundraising; educational resources and documentation reducing learning barriers; and community spaces for developers to share knowledge and collaborate.

Developer ecosystem components include:

- **Developer Grant Program**: Funded by the community treasury with a diversified grant structure: microgrants (up to $10K) for individuals building proof-of-concepts, project grants ($50K–$500K) for teams building production applications, and infrastructure grants for critical ecosystem tooling (indexers, analytics, wallets, bridges).

- **Hackathons and Builder Programs**: Community-organised and sponsored hackathons targeting developers from Ethereum, Cosmos, and Solana ecosystems. The competitions showcase QoreChain's unique capabilities (cross-chain atomicity, quantum-safe cryptography, AI-native protocol) and attract talented developers to explore new application possibilities.
- **Technical Documentation and Education**: Comprehensive documentation covering the protocol design, API specifications, and development best practices for all three VMs. Video tutorials, interactive examples, and sandbox environments enable developers to learn without establishing local testnets. Sample applications (DEX, NFT collection, governance DAO) serve as reference implementations demonstrating best practices.
- **Community Governance**: By this phase the community governance system is expected to be fully operational with active proposal submission, voting, and execution. Developer community members are intended to participate in governance decisions affecting protocol parameters, treasury allocation, and feature prioritisation, establishing developer voices in protocol evolution.
- **Developer Support Infrastructure**: Official developer forums, Discord channels, and office hours provide dedicated support for developers building applications. Expert developers from the core team provide code reviews, architecture consulting, and performance optimisation guidance for community projects.

14.10 Phase 8: Enterprise and Cross-Chain Maturity (2028)

Phase 8 targets full ecosystem maturity with enterprise-grade production deployments and comprehensive cross-chain coverage.

14.10.1 Enterprise Production Deployments

Phase 8 is planned to transition from pilot programs to production deployments across regulated industries, establishing QoreChain as critical financial infrastructure. Phase 8 is intended to represent the full maturity of QoreChain's strategic vision: operating as the quantum-safe settlement layer for global financial systems, supporting institutional-scale digital asset value, and processing settlement transactions for institutions whose failure would have global economic consequences. Production deployments will operate under the scrutiny of financial regulators, requiring compliance with securities regulations, anti-money laundering standards, and prudential requirements for custodial operations. Enterprise deployments are intended to justify the security, reliability, and operational sophistication investments made throughout the development roadmap.

Phase 8 production deployments are intended to establish QoreChain's critical role across regulated industries:

- **CBDC Settlement Rails**: Production wholesale and retail CBDC infrastructure is envisioned for central bank partners spanning multiple jurisdictions. Wholesale CBDCs for inter-bank settlement are designed to enable same-day settlement with PQC-signed finality proofs suitable for regulatory reporting. Retail

CBDCs for consumer transactions are intended to demonstrate programmable money (e.g., spending restrictions, expiry dates), automated tax collection, and cross-border payment settlement in minutes.

- **Securities Tokenization**: Regulated tokenised bond and equity platforms are planned with T+0 (trade-plus-zero-days) atomic settlement. Securities on QoreChain are designed to support programmable dividends (automatically distributed to token holders), shareholder voting through smart contracts, and atomic settlement of complex multi-leg trades that previously required days of clearing.

- **Trade Finance Automation**: Production implementations of letters of credit, purchase order financing, and supply chain finance are planned on CosmWasm. These are designed to automate documentary proof checks, escrow release, and payment settlement, targeting meaningful reduction in trade finance costs and enabling participation by small and medium enterprises previously priced out of traditional trade finance.

- **Institutional Custody**: Targeted partnerships with qualified tier-1 institutional custodians for PQC-native digital asset custody, subject to regulatory qualification and commercial agreement. Such custodians would operate QoreChain validators, store digital assets on behalf of institutional clients, and provide custody proofs signed with ML-DSA-87, offering institutional-grade security guarantees aligned with their practices for traditional assets.

These production deployments are intended to represent the culmination of QoreChain's design objectives: quantum-safe settlement infrastructure enabling institutions to confidently transact and settle with cryptographic guarantees extending decades into the future.

14.10.2 Full Bridge Coverage

By Phase 8, all 25 QCB bridge endpoints (8 IBC + 17 QCB) are planned to operate at production capacity with full bridge validator coverage. This is intended to complete QoreChain's One-Hop-Swap topology: every pair of connected chains would execute atomic transfers through QoreChain with exactly one intermediate hop, positioning QoreChain as the de facto routing hub for cross-chain value movement.

QCAI route optimisation is intended to operate autonomously across all IBC and QCB routes, providing real-time fee/speed/security optimisation for every cross-chain transfer. The routing engine is designed to learn from accumulated network data across testnets and Phase 7 pilot deployments, enabling statistically informed routing decisions intended to outperform user-specified preferences (fee minimisation, speed optimisation, security maximisation).

Full bridge coverage is intended to enable new application patterns: liquidity aggregators could seamlessly route trades across all connected chains; yield farming protocols could deploy capital wherever yields are highest; and cross-chain MEV-resistant applications could utilise QoreChain's multiple routing options to prevent frontrunning attacks by splitting transactions across non-overlapping routes. Bridge validator sets are planned to reach broad geographic distribution, ensuring that no single jurisdiction

or cloud provider controls critical bridge infrastructure. The complete validator network is designed to target sub-second attestation confirmation even for geographically dispersed nodes. All bridge state commitments are PQC-signed, providing cryptographic certainty that bridge consensus remains secure against quantum attacks. The routing optimization engine is designed to ingest cross-chain transaction data from the public testnet and Phase 7 pilot deployments, building statistical models of gas prices, swap depths, and relative latencies across all 25 connected chains. These models are intended to power real-time routing decisions that target 5-10 percent execution improvement over static routes.

14.10.3 QCAI Model Governance

On-chain governance for QCAI model updates, enabling the community to vote on AI model upgrades, parameter changes, and new capability additions through the standard proposal lifecycle. This addresses a fundamental challenge with AI systems in blockchain: who controls the models, how do model updates happen, and how can the community ensure AI systems remain aligned with protocol interests?

The QCAI Model Governance framework operates as follows: (1) researchers or community members propose new QCAI models via governance proposals, including benchmarks demonstrating the model's performance improvement and formal analysis of changes to anomaly detection false positive/false negative rates; (2) the community votes on the proposal through standard governance mechanisms (Tier 2 for parameter tuning, Tier 1 for major architectural changes); (3) approved models are staged to testnet, operated in shadow mode (logging but not enforcing decisions) for a 1–2 month period to collect real-world performance data; (4) after community review of testnet performance, a second governance proposal activates the model on mainnet.

This governance framework ensures that QCAI evolution remains transparent and community-controlled, preventing unilateral control by any developer or team. The framework also establishes a precedent for AI governance in blockchain systems: AI components are powerful tools that require explicit, transparent, community-approved activation rather than silent background deployments. Governance proposals for new QCAI models include statistical analysis of false positive and false negative rates under simulated attack conditions, ensuring that community voters understand the security trade-offs of any proposed change. The shadow mode testing period on testnet provides real-world performance data before mainnet activation; the community can analyze logs of QCAI decisions, understand false positive rates, and assess whether the proposed model improves protocol security. If shadow mode results show unacceptable performance, the governance proposal is withdrawn and the development team returns to refinement. This transparent iterative process builds community confidence that QCAI models are security tools, not black boxes controlled by developers.

14.10.4 Algorithm Agility Expansion

As post-quantum cryptographic standards evolve beyond NIST's initial 2024 finalisation, the Algorithm Agility Framework enables governance-controlled addition of new algorithms without breaking existing protocols. This future-proofs QoreChain against the evolution of both quantum threats and cryptographic knowledge.

Potential algorithm additions as they reach standardisation and community consensus include:

- **Additional NIST-Standardised Signature Schemes**: As NIST publishes additional post-quantum standards beyond ML-DSA and ML-KEM, governance can approve inclusion of these algorithms as alternatives. Multiple signature schemes provide redundancy: even if a flaw is discovered in one algorithm family, the network continues operating with proven alternatives.
- **Hash-Based Signature Schemes (SPHINCS+)**: These schemes provide provable security guarantees independent of unproven assumptions (unlike lattice-based schemes which rely on the hardness of Learning With Errors). SPHINCS+ offers broader security assurance at the cost of larger signature sizes, suitable for applications valuing provable security over performance.
- **Isogeny-Based and Code-Based Algorithms**: Alternative PQC families provide diversity. As these algorithm families mature and reach standardisation, including them as options provides additional security hedging against any unforeseen vulnerabilities in lattice-based schemes.

The framework ensures that QoreChain remains secure even as cryptographic knowledge evolves. By 2030 and beyond, as quantum computing capabilities become clearer and new cryptographic research emerges, QoreChain's community can approve algorithm additions through governance without hard forks or protocol disruption.

14.10.5 Guardian Deprecation

As validator governance participation matures and the network demonstrates sustained high-quorum governance activity, the guardian multisig threshold is progressively raised and eventually deprecated through a Constitutional (Tier 1) governance proposal, completing the decentralisation trajectory. This planned deprecation reflects QoreChain's commitment to decentralisation: the guardian multisig is a temporary bridge to full community governance rather than a permanent centralised control mechanism.

The deprecation path follows three stages: **Stage 1 (Month 1 to 12)**: Guardian operates with 5-of-7 quorum (two-thirds plus one), making it difficult to unilaterally deploy changes but still accessible for genuine emergencies. **Stage 2 (Month 13 to 24)**: Guardian threshold increases to 6-of-7 (nearly unanimous), ensuring that guardian actions reflect strong consensus among multisig members. **Stage 3 (Month 25+)**: Upon demonstration of sustained high-quorum governance activity (70 percent or greater participation on security-relevant proposals), a Constitutional proposal deprecates the guardian entirely, transitioning full emergency response authority to the expedited Tier 3 governance process.

The planned deprecation provides transparency: community members know from genesis that the guardian multisig is temporary, incentivising the development of efficient emergency governance processes that will eventually replace it. The staged threshold increases ensure that the transition is gradual, allowing the community to gain confidence in emergency governance before relying on it exclusively. Throughout the deprecation path, guardian actions remain subject to full transparency; every use of guardian authority is logged and published in the weekly governance digest. This

record allows the community to audit guardian decisions, challenge abuses of authority, and build confidence that the guardian operates only for legitimate emergencies rather than routine governance. As Tier 3 expedited governance becomes more frequently used for emergency responses, community members develop familiarity and confidence with the process. By the time the guardian deprecates, the network has months of experience with fast-path governance responding to actual emergencies, ensuring a smooth transition to fully decentralised emergency response.

14.11 Long-Term Vision (2029 and Beyond)

QoreChain's long-term trajectory is defined by the intersection of quantum computing advancement, AI capability growth, and global regulatory evolution. The vision extends beyond today's blockchain landscape into a future where quantum threat and advanced AI are both fully materialised facts, reshaping how digital infrastructure operates.

Post-Quantum Ecosystem Maturity. As cryptographically relevant quantum computers approach viability (projected early-to-mid 2030s), QoreChain's position as the established quantum-safe settlement layer becomes increasingly strategic. Networks that deferred PQC migration will face urgent, disruptive transitions: they must hard-fork to new cryptographic schemes, migrate all existing keys, and somehow maintain trust continuity through the transition. QoreChain's users and applications are already protected through the hybrid migration path described in Phase 3, having transitioned gradually to PQC-only operation as the quantum threat materialized. The cost advantage of early migration becomes evident as late movers scramble to catch up.

Quantum Key Distribution Readiness. As QKD (Quantum Key Distribution) networks become commercially viable through initiatives like the European Quantum Internet Alliance and US National Quantum Initiative, QoreChain's infrastructure provisions accommodate integration of physics-based key exchange. QKD networks provide provable information-theoretic security against quantum adversaries, adding a second layer of quantum security beyond the mathematical protections of lattice-based cryptography. QoreChain's algorithm agility framework enables seamless integration of QKD-derived keys once commercial networks achieve the bandwidth and cost profile required for blockchain-scale deployment.

Decentralised AI Model Governance. The evolution of on-chain AI governance toward fully decentralised model selection, training data curation, and inference verification enables the network to collectively manage its intelligence layer without centralised control. This vision encompasses: governance-voted model training on decentralised datasets; verification of model properties through formal methods and differential privacy analysis; and incentive mechanisms rewarding improvements to QCAI performance and security. The network becomes not just a smart contract platform but an entity with collective learning and adaptive intelligence.

Global CBDC Interoperability Hub. QoreChain is positioned as the quantum-safe clearing layer connecting sovereign CBDC systems worldwide through the One-Hop-Swap topology. As central banks issue their own digital currencies (e.g., Fed Now, e-Euro, digital yuan), QoreChain's infrastructure enables frictionless cross-border digital currency settlement. A user in Switzerland can seamlessly convert CHF digital francs to USD digital dollars for international payment in a single atomic transaction, with final settlement in seconds rather than days.

Dominant Quantum-Safe Settlement Layer. The strategic objective is QoreChain as the foundational settlement infrastructure for the post-quantum digital economy, securing trillions of dollars in digital assets, cross-chain liquidity, and institutional value against the quantum threat. This vision acknowledges that the blockchain ecosystem has evolved from fringe technology to critical infrastructure; QoreChain's mission is to ensure this infrastructure remains secure even in a post-quantum computing era where classical cryptography is obsolete.

14.12 Milestone Dependency Analysis

The development roadmap contains critical dependencies between milestones. The following analysis identifies the critical path and quantifies delivery risk.

14.12.1 Critical Path

The critical path through the roadmap is the longest sequence of dependent milestones that determines the minimum time to mainnet launch:

$$\text{CP} = \text{Multi-node testnet} \rightarrow \text{Benchmarks} \rightarrow \text{Security audits} \rightarrow \text{Community testnet} \rightarrow \text{Validation gate} \rightarrow \text{Mainnet} \tag{14.1}$$

Each milestone on the critical path must complete before the next can begin. Delays on the critical path directly delay mainnet launch; delays on non-critical-path milestones have float. The multi-node testnet deployment represents the bottleneck in schedule; this phase requires coordination across distributed validator infrastructure, resolution of network-layer issues that only emerge in multi-node configurations, and hardening of consensus mechanisms under realistic latency and Byzantine conditions. Security audits run in parallel and must complete before community testnet, ensuring that discovered vulnerabilities can be patched before public exposure. The community testnet 30-day run serves dual purposes: (1) validation that the mainnet-ready codebase functions correctly over extended periods, and (2) socialization of the network to early adopters and ecosystem partners, building a critical mass of informed community members before mainnet launch. The validation gate review is deliberately short; it represents a formal approval ceremony where the engineering team presents evidence that all phase objectives were met and makes the recommendation to proceed to mainnet preparation. Mainnet preparation includes final documentation, launch playbook development, ecosystem onboarding, and contingency planning for day-one issues.

14.12.2 PERT Estimation

Using the Program Evaluation and Review Technique (PERT), each critical-path milestone m_i is estimated with optimistic (a_i), most likely (m_i), and pessimistic (b_i) durations in weeks:

$$\hat{T}_i = \frac{a_i + 4m_i + b_i}{6}, \qquad \sigma_i^2 = \left(\frac{b_i - a_i}{6}\right)^2 \tag{14.2}$$

The expected total critical-path duration:

Table 14.1: PERT estimates for critical-path milestones (weeks)

Milestone	a_i	m_i	b_i	$\hat{T}_i$	σ_i
Multi-node testnet deployment	2	3	6	3.3	0.67
Performance benchmarking	1	2	4	2.2	0.50
Security audits (3 parallel)	4	6	10	6.3	1.00
Community testnet (30-day run)	4	5	7	5.2	0.50
Validation gate review	1	1	2	1.2	0.17
Mainnet preparation	2	3	5	3.2	0.50

$$\hat{T}_{\text{total}} = \sum_i \hat{T}_i = 21.4 \text{ weeks} \tag{14.3}$$

with combined standard deviation:

$$\sigma_{\text{total}} = \sqrt{\sum_i \sigma_i^2} = \sqrt{0.45 + 0.25 + 1.00 + 0.25 + 0.03 + 0.25} = 1.49 \text{ weeks} \tag{14.4}$$

14.12.3 Monte Carlo Delivery Estimate

Assuming each milestone duration follows a Beta-PERT distribution, the probability of completing the critical path within a given number of weeks W is approximated by the normal distribution:

$$P(T_{\text{total}} \leq W) = \Phi\left(\frac{W - \hat{T}_{\text{total}}}{\sigma_{\text{total}}}\right) \tag{14.5}$$

For a Q2 2026 mainnet launch with Phase 5 beginning Q1 2026 (approximately 26 weeks available):

$$P(T_{\text{total}} \leq 26) = \Phi\left(\frac{26 - 21.4}{1.49}\right) = \Phi(3.09) \approx 0.999 \tag{14.6}$$

The 26-week window covers the expected critical path with 4.6 weeks of float, supporting the Q2 2026 target with high confidence. Even at the 99th percentile pessimistic estimate ($\hat{T} + 2.33\sigma = 24.9$ weeks), the schedule retains positive margin.

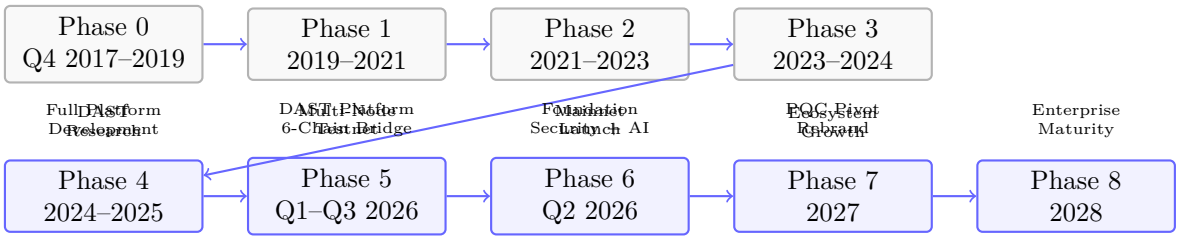

Figure 14.1: QoreChain development roadmap from initial research (Q4 2017) through enterprise maturity (2028). Grey boxes represent completed historical phases; blue boxes represent current and future phases.

Chapter 15

Technical Specifications

This chapter consolidates QoreChain's technical parameters into a comprehensive reference. All values represent the protocol's designed specifications as of June 2026. Parameters marked as governance-configurable may be modified through the on-chain governance process described in Chapter 12.

15.1 Network Parameters

The network parameters define the foundational operational characteristics of QoreChain, including chain identification, tokenomics primitives, address formatting, and consensus timing parameters. These parameters establish the baseline constraints within which all protocol operations occur. The parameters are split into immutable protocol constants (chain ID, native token, address prefixes) and governance-configurable parameters that may be adjusted via on-chain governance proposals.

The block production system is governed by a target block time parameter, typically set to 6 seconds. This interval defines the expected rate of canonical chain progression and informs derived parameters such as epoch length and unbonding duration calculations. The network maintains a bounded validator set with a minimum of 4 active validators (to prevent trivial consensus) and a maximum of 150 validators by default, though this upper bound is adjustable through governance.

The token denomination system uses the QOR as the primary unit, with the smallest divisible denomination being the microQOR (uqor), where 1 QOR equals 1,000,000 uqor. This structure enables both large transactions (denominated in QOR) and precise fee accounting (denominated in uqor), mirroring the successful denomination scheme used in other blockchain platforms.

The unbonding period of 21 days provides a security window during which delegators retain ability to revoke their stake before it enters the withdrawal queue. This period serves as a secondary defense against validator misbehavior, allowing delegators to respond to observed misconduct with capital loss recovery. The epoch length parameter (100 blocks) aligns with the emission and reward distribution schedules described in Section 15.7.

$$\text{Block time variance} = \frac{\sigma_{\text{block}}}{T_{\text{target}}} \tag{15.1}$$

where σ_{block} is the standard deviation of observed block production intervals and $T_{\text{target}} = 6$ seconds is the target. Network parameters are designed for variance targets

below 0.15 (15% coefficient of variation).

Table 15.1: Core network parameters

Parameter	Value	Configurable
Chain ID (testnet)	`qorechain-diana`	No
Native token	QOR	No
Smallest denomination	uqor (1 QOR = 10^6 uqor)	No
Account prefix (Bech32)	`qor`	No
Validator operator prefix	`qorvaloper`	No
Consensus address prefix	`qorvalcons`	No
Target block time	6 seconds	Governance (Tier 1)
Minimum validators	4	Governance (Tier 1)
Maximum validators	150 (default)	Governance (Tier 1)
Epoch length	100 blocks	Governance (Tier 2)
Unbonding period	21 days	Governance (Tier 1)
Maximum gas per block	Governance-set	Governance (Tier 1)
Transaction memo max length	256 bytes	Governance (Tier 2)

```
# Query current network parameters
qorechaind query params subspace baseapp BlockParams --output json

# Query chain status
qorechaind status --output json

# Example response (partial):
# {
#   "node_info": {
#     "network": "qorechain-diana",
#     "version": "2.6.0"
#   },
#   "sync_info": {
#     "latest_block_height": "5240100",
#     "latest_block_time": "2026-03-22T14:30:06Z"
#   }
# }
```

The governance-configurable parameters enable protocol evolution without requiring new binary releases. Tier 1 governance (constitutional changes) controls critical parameters such as target block time, minimum/maximum validator counts, and unbonding duration. Tier 2 governance (standard parameters) controls operational parameters such as epoch length and transaction memo size limits. This separation ensures that fundamental security constraints remain under elevated governance thresholds, while less critical operational parameters can be adjusted with broader stakeholder approval.

The maximum gas per block parameter is expressed as a governance-set value to provide flexibility in adapting to network conditions. As computational capacity increases or protocol sophistication evolves, validators can collectively agree to adjust the per-block gas allowance without breaking protocol compatibility. This parameter

directly influences transaction throughput and fee structures, making it a critical lever for network performance tuning.

Network parameters are queried and set through a unified parameter subspace system, allowing operators to introspect configuration at runtime and propose changes through governance. The implementation maintains backward compatibility with prior parameter values, enabling gradual rollouts of parameter changes across the validator set.

15.2 Cryptographic Specifications

15.2.1 Post-Quantum Algorithm Parameters

QoreChain's cryptographic foundation is built on four NIST-standardised post-quantum algorithms, selected from the FIPS 203, 204, and 205 standards finalised in 2024. These algorithms replace classical elliptic curve and RSA cryptography with schemes believed secure against both classical and quantum adversaries. The selection prioritizes algorithms with longer deployment history, stronger security reductions, and practical performance characteristics.

ML-DSA-87 (Module-Lattice-Based Digital Signature Algorithm at security level 5) serves as the primary signature algorithm, replacing ECDSA-256. The parameter set 87 indicates the primary dimension used in the underlying lattice problem, targeting 256-bit classical security and 128-bit quantum security. ML-KEM-1024 provides key encapsulation for symmetric key agreement, used in protocols requiring forward secrecy. SLH-DSA-256f provides hash-based signatures as a fallback mechanism, offering extreme simplicity and conservatively-proven security but with larger signature sizes.

The mathematical security of these algorithms rests on different hard problems than classical cryptography. ML-DSA and ML-KEM both rely on the Module-Learning With Errors (Module-LWE) problem, a variant of the Learning With Errors lattice problem that has remained unsolved since its introduction in 2005. SLH-DSA derives security from collision-resistance of cryptographic hash functions. These distinct mathematical foundations provide defense-in-depth against undiscovered attacks.

$$\begin{aligned}\text{Security margin} &= \min(S_{\text{ML-DSA}}, S_{\text{ML-KEM}}, S_{\text{SLH-DSA}}) \\ &= \min(256 \text{ bits}, 256 \text{ bits}, 256 \text{ bits}) = 256 \text{ bits} \end{aligned} \tag{15.2}$$

The security margin represents the smallest security claim across the cryptographic primitives, ensuring that the weakest link in the cryptographic chain bounds the overall security.

Table 15.2: PQC algorithm specifications

Algorithm	NIST Standard	Public Key	Signature/CT	Security (classical)	Security (quantum)
ML-DSA-87	FIPS 204	2,592 B	4,627 B	256-bit	128-bit
ML-KEM-1024	FIPS 203	1,568 B	1,568 B	256-bit	128-bit
SLH-DSA-256f	FIPS 205	64 B	49,856 B	256-bit	128-bit
SHAKE-256	FIPS 202	N/A	32 B (digest)	256-bit	128-bit

Code examples demonstrating PQC key generation and verification:

```
# Generate a new ML-DSA-87 keypair for signing
qorechaind keys add mykey --algo ml-dsa-87

# Query public key information
qorechaind keys show mykey --output json

# Example output structure:
# {
#   "name": "mykey",
#   "type": "local",
#   "address": "qor1abc...def",
#   "pubkey": {
#     "@type": "/cosmos.crypto.ml_dsa_87.PubKey",
#     "key": "<base64-encoded-2592-byte-public-key>"
#   },
#   "mnemonic": ""
# }

# Verify signature using PQC precompile
qorechaind query pqc verify-signature \
  --message "test message" \
  --signature "<signature-bytes>" \
  --pubkey "<pubkey-bytes>"
```

15.2.2 Comparison with Classical Cryptography

The transition from classical ECDSA-based signatures to post-quantum algorithms introduces measurable overhead in multiple dimensions: key sizes, signature sizes, signing time, and verification time. Understanding these tradeoffs is essential for evaluating the practical impact of quantum-safe cryptography on blockchain throughput and user experience.

Classical ECDSA-256 (elliptic curve digital signature algorithm over the NIST P-256 curve) has dominated blockchain signature schemes due to its compact 33-byte compressed public keys and 64-byte signatures, combined with sub-millisecond signing and verification on modern processors. However, ECDSA is vulnerable to Shor's algorithm running on a sufficiently large quantum computer, reducing its security margin to approximately zero in the presence of a cryptographically-relevant quantum computer (CRQC).

ML-DSA-87 trades signature size for quantum resistance. The 2,592-byte public key represents a 78.5x increase over ECDSA-256, while the 4,627-byte signature represents a 72.3x increase. These larger structures impose overhead on network bandwidth, storage, and transaction size. However, the verification time remains competitive at approximately 3x slower than ECDSA, while signing time is approximately 16x slower. This signing overhead is primarily a client-side concern, as signature generation happens off-chain during transaction preparation, while verification happens on-chain and benefits from batch verification optimizations.

The quantum security comparison is asymmetric in QoreChain's favour. ML-DSA-87 provides 128 bits of quantum security, meaning the best known quantum attack

requires approximately 2^{128} quantum gate operations. By contrast, ECDSA-256 provides 0 bits of quantum security (it is broken completely by Shor's algorithm). For assets with long security horizons (multiple decades), quantum resistance is not merely an incremental improvement but a categorical difference.

$$\text{Network overhead} = \frac{(|pk_{\text{ML-DSA}}| + |\sigma_{\text{ML-DSA}}|) - (|pk_{\text{ECDSA}}| + |\sigma_{\text{ECDSA}}|)}{|pk_{\text{ECDSA}}| + |\sigma_{\text{ECDSA}}|} = \frac{7122 - 97}{97} = 7244\% \quad (15.3)$$

This overhead is per transaction and represents the additional bytes consumed by quantum-safe signatures. For a blockchain targeting 5,000 TPS with average transaction size of 300 bytes, the PQC overhead translates to approximately 35.6 MB/s of additional bandwidth, which is well within the capacity of modern data center networks.

Table 15.3: QoreChain PQC vs classical ECDSA-256

Property	**ECDSA-256**	**ML-DSA-87**	**Factor**
Public key size	33 B	2,592 B	78.5x
Signature size	64 B	4,627 B	72.3x
Sign time (x86, 3.5 GHz)	∼0.05 ms	∼0.8 ms	16x
Verify time (x86, 3.5 GHz)	∼0.10 ms	∼0.3 ms	3x
Quantum security	0 bits	128 bits	∞

15.2.3 Concrete Security Estimates

The security of ML-DSA-87 against quantum attack is bounded by the hardness of the Module-SIS and Module-LWE problems. The best known quantum attacks require:

$$T_{\text{quantum}}(\text{ML-DSA-87}) \geq 2^{128} \text{ quantum gate operations} \quad (15.4)$$

By comparison, breaking ECDSA-256 via Shor's algorithm requires:

$$T_{\text{quantum}}(\text{ECDSA-256}) \approx 2^{64} \text{ quantum gate operations} \quad (15.5)$$

The security margin of ML-DSA-87 over ECDSA-256 against quantum adversaries:

$$\Delta_{\text{security}} = \frac{T_{\text{quantum}}(\text{ML-DSA-87})}{T_{\text{quantum}}(\text{ECDSA-256})} = \frac{2^{128}}{2^{64}} = 2^{64} \approx 1.84 \times 10^{19} \quad (15.6)$$

15.2.4 Algorithm Agility

The PQC Algorithm Agility Framework supports governance-controlled algorithm management:

Table 15.4: Algorithm agility parameters

Parameter	Value
Maximum concurrent algorithms	5
Algorithm addition governance tier	Tier 1 (Constitutional)
Algorithm deprecation governance tier	Tier 1 (Constitutional)
Deprecation grace period	180 days (minimum)
Hybrid signature support	Classical + PQC dual signatures
Migration phases	3 (Classical, Hybrid, PQC-Only)

15.3 Consensus Parameters

15.3.1 Combined Proof of Stake (CPoS)

Combined Proof of Stake (CPoS) extends traditional Delegated Proof of Stake (DPoS) by adding reputation-based and stake-based validator selection mechanisms. Rather than using a single metric (stake amount) to rank validators, CPoS synthesizes three independent validator pools, each selected using different criteria, and combines their consensus weight according to fixed proportions.

The motivation for multi-pool consensus stems from limitations in pure stake-weighted systems: large token holders can dominate consensus even without operating high-quality infrastructure, while small but highly-reliable operators are disadvantaged. CPoS decouples validator selection from capital concentration by creating distinct pathways to consensus participation.

The Reputation PoS (RPoS) pool contains the top 25th percentile of validators by reputation score (a composite metric of uptime, governance participation, and historical slashing). The Delegated PoS (DPoS) pool contains the top 25th percentile of validators by delegation ratio (stake delegated to the validator divided by validator's own stake). The Standard PoS pool contains all remaining active validators. These pools are recomputed each epoch based on updated metrics.

The consensus weight allocation formula ensures that reputation and delegation capacity contribute equally ($40\% + 35\% = 75\%$) while maintaining at least 25% participation from the broader validator set. This structure prevents excessive concentration in any single dimension while preserving the economic security property of traditional PoS.

$$W_v = 0.40 \cdot \mathbb{1}[\text{RPoS pool}] + 0.35 \cdot \mathbb{1}[\text{DPoS pool}] + 0.25 \cdot \mathbb{1}[\text{PoS pool}] \tag{15.7}$$

where $\mathbb{1}[\cdot]$ is an indicator function. Note that a validator can belong to multiple pools simultaneously if its metrics qualify for membership.

Code examples for querying CPoS pool membership and consensus weight:

```
# Query all active validators and their pool assignments
qorechaind query staking validators --output json

# Query specific validator CPoS pool membership
qorechaind query consensus pool-membership \
  qorvaloper1abc...def --output json
```

Table 15.5: CPoS pool configuration

Pool	Weight	Classification Criterion
Reputation PoS (RPoS)	0.40	Top 25th percentile reputation score
Delegated PoS (DPoS)	0.35	Top 25th percentile delegation ratio
Standard PoS (PoS)	0.25	Remaining validators

```
# Example response:
# {
#   "validator": "qorvaloper1abc...def",
#   "pools": ["rpos", "dpos"],
#   "consensus_weight": 0.75,
#   "rpos_weight": 0.40,
#   "dpos_weight": 0.35,
#   "epoch": 12450,
#   "next_recompute_epoch": 12550
# }

# Query pool statistics at current epoch
qorechaind query consensus pool-stats --output json
```

15.3.2 Reputation Scoring

Reputation scoring provides a quantitative measure of validator quality based on operational metrics including uptime, block production participation, and governance engagement. Unlike stake-based selection, reputation scoring rewards validators for consistent, high-quality operation over time. The reputation system uses real-time on-chain metrics collected during each epoch, ensuring transparency and eliminating off-chain reputation services.

The reputation score for validator i at epoch t is computed as a weighted sum of three components:

$$r_i^{(t)} = \omega_{\text{up}} \cdot u_i^{(t)} + \omega_p \cdot p_i^{(t)} + \omega_v \cdot v_i^{(t)} - \text{penalty}(s_i^{(t)}) \tag{15.8}$$

where $u_i^{(t)}$ is the validator's uptime fraction (blocks included divided by total blocks), $p_i^{(t)}$ is the block-signing participation rate, $v_i^{(t)}$ is the governance vote participation rate, and $s_i^{(t)}$ is any slashing penalty incurred in epoch t. The weights are set to $\omega_{\text{up}} = 0.40$, $\omega_p = 0.35$, and $\omega_v = 0.25$.

To prevent reputation from becoming stale or permanently damaged by isolated incidents, the system applies exponential decay over a rolling window of epochs:

$$\bar{r}_i^{(t)} = \frac{\sum_{e=t-W+1}^{t} \lambda^{t-e} \cdot r_i^{(e)}}{\sum_{e=t-W+1}^{t} \lambda^{t-e}} \tag{15.9}$$

where $\lambda = 0.95$ is the decay factor (reflecting 95% weight on the current epoch and 5% cumulative weight on prior epochs) and W is the window size (typically 100 epochs). This rolling window ensures that a single slashing event impacts the reputation score

heavily in the near term but gradually decays, allowing validators to rebuild reputation through consistent good performance.

Table 15.6: Reputation scoring parameters

Parameter	Symbol	Default
Uptime weight	ω_u	0.40
Block production weight	ω_p	0.35
Vote participation weight	ω_v	0.25
Decay factor	λ_R	0.95
Reputation range	$[r_{\min}, r_{\max}]$	$[0, 1]$

Code examples for querying reputation scores and computing components:

```
# Query reputation score for a validator
qorechaind query reputation score \
  qorvaloper1abc...def --output json

# Example response:
# {
#   "validator": "qorvaloper1abc...def",
#   "reputation_score": 0.92,
#   "uptime_fraction": 0.98,
#   "block_production": 0.95,
#   "governance_participation": 0.85,
#   "slashing_penalty": 0.00,
#   "epoch": 12450,
#   "window_size_epochs": 100,
#   "decay_factor": 0.95
# }

# Query reputation leaderboard
qorechaind query reputation leaderboard --limit 20 --output json

# Calculate expected reputation impact of governance participation
qorechaind query reputation calc-participation-impact \
  --current-participation 0.80 \
  --new-participation 0.90 \
  --output json
```

15.3.3 Bonding Curve

The bonding curve mechanism determines the block rewards allocated to validators as a function of their stake amount, operational longevity, reputation score, and time. The curve is designed to incentivize long-term participation and high-quality operation while maintaining economic self-sufficiency for validators.

The bonding curve formula combines four reward factors: a base coefficient reflecting the epoch's total emission, the validator's stake amount, a logarithmic longevity scaling that rewards long-standing validators without creating compounding advan-

tages, and a reputation multiplier that amplifies or diminishes rewards based on past performance.

The longevity component uses logarithmic scaling to prevent unbounded reward acceleration. A validator with 1 QOR stake gains approximately 10.5% additional reward if they have been active for 1,000 epochs versus 100 epochs (using $\alpha = 0.1$ and the difference $\ln(1+1000) - \ln(1+100) \approx 0.105$). This design rewards stability without creating exponential advantages for age.

The reputation multiplier $Q(r_v)$ ranges from 0.5 (for validators with reputation score near 0) to 2.0 (for validators with reputation score near 1), creating a 4x spread in rewards based purely on quality differential. This ensures that the reputation system has material impact on validator profitability.

The time decay function $P(t)$ captures epoch-specific dynamics such as total stake in the network, total emission rate, and network utilization. It typically decreases over the protocol's lifespan as inflation diminishes, reducing per-block rewards even as validator participation increases.

$$R(v,t) = \beta \cdot S_v \cdot (1 + \alpha \cdot \ln(1 + L_v)) \cdot Q(r_v) \cdot P(t) \tag{15.10}$$

Table 15.7: Bonding curve default parameters

Parameter	**Symbol**	**Default**
Base reward coefficient	β	1.0
Longevity scaling factor	α	0.1
Reputation multiplier range	$Q(r)$	$[0.5, 2.0]$
Time decay function	$P(t)$	Epoch-aligned decay

Code examples for calculating bonding curve rewards:

```
# Calculate expected block reward for a validator
qorechaind query rewards calc-reward \
  --validator qorvaloper1abc...def \
  --stake 100000uqor \
  --reputation-score 0.95 \
  --longevity-epochs 1000 \
  --output json

# Example response:
# {
#   "base_coefficient": 1.0,
#   "stake_component": 100.0,
#   "longevity_multiplier": 1.105,
#   "reputation_multiplier": 1.950,
#   "time_decay": 0.95,
#   "estimated_block_reward": "208.73uqor",
#   "expected_annual_yield": "0.042"
# }

# Query current epoch decay parameters
qorechaind query rewards epoch-params --output json
```

```
# Simulate reward under different reputation scenarios
qorechaind query rewards simulate-reputation-impact \
  --base-reward 100uqor \
  --reputation-range 0.5,0.75,1.0 \
  --output json
```

15.3.4 Progressive Slashing

Progressive slashing implements graduated penalties for validator misbehavior, scaling the severity of punishment based on the frequency and recency of offenses. Unlike simple fixed-penalty slashing, progressive slashing recognizes that occasional isolated failures are different from systematic misbehavior and penalizes accordingly.

The slashing system distinguishes between two primary offense types: downtime (missed block proposals or failures to participate in consensus) and double-signing (equivocating by signing two conflicting blocks). Double-signing represents more serious misbehavior as it demonstrates intentional Byzantine activity, while downtime can result from temporary infrastructure failures.

For downtime offenses, each missed block incurs a small per-block penalty (0.01% of stake), accumulated over the jail period. The base penalty of 1% applies only after a validator exceeds a threshold of missed blocks within a rolling window. Multiple downtime offenses within a short time window trigger escalation: the second offense costs 2% of stake, the third costs 4% (2x), up to the maximum cap of 33% of stake.

The temporal half-life parameter (100,000 blocks, approximately 7 days) creates a decay mechanism whereby old offenses gradually lose influence on the escalation count. After 100,000 blocks, an offense contributes only half its weight to the escalation counter, and after 200,000 blocks it contributes only 1/4. This design allows validators to demonstrate reformation: if a validator commits an offense, serves the jail period with perfect behavior, and waits for the half-life window to elapse, they can reduce their escalation multiplier.

For double-signing, the penalty is immediate and severe (5% of stake base), with the validator jailed permanently until governance explicitly votes to unjail them. This creates strong deterrence against intentional Byzantine behavior while preserving the possibility of remediation through governance processes.

$$\text{Slashing penalty} = \min(p_{\text{base}} \cdot (1 + \text{esc}^{k-1}), P_{\text{max}}) \tag{15.11}$$

where k is the number of offenses within the effective window (accounting for half-life decay).

Code examples for querying and simulating slashing penalties:

```
# Query slashing history for a validator
qorechaind query slashing signing-info \
  qorvalcons1abc...def --output json

# Example response:
# {
#   "address": "qorvalcons1abc...def",
#   "start_height": 1,
```

Table 15.8: Progressive slashing parameters

Parameter	Symbol	Default
Base penalty	p_{base}	1% of stake
Escalation factor	esc	2.0 per effective count
Temporal half-life	$H_{1/2}$	100,000 blocks (≈7 days)
Maximum penalty cap	$P_{\max}$	33% of stake
Downtime penalty	–	0.01% per missed block
Double-sign base penalty	–	5% of stake
Jail duration (downtime)	–	600 blocks (≈1 hour)
Jail duration (double-sign)	–	Permanent (governance unjail)

```
#   "index_offset": 5000000,
#   "jailed_until": "0001-01-01T00:00:00Z",
#   "tombstoned": false,
#   "missed_blocks_counter": 0
# }

# Query parameters for progressive slashing
qorechaind query slashing params --output json

# Simulate escalation penalty calculation
qorechaind query slashing calc-penalty \
  --base-stake 1000000uqor \
  --num-offenses 3 \
  --blocks-since-last-offense 50000 \
  --half-life-blocks 100000 \
  --output json

# Query jailed validators and unjail deadline
qorechaind query slashing signing-infos | grep -A2 jailed_until
```

15.3.5 Reinforcement Learning Agent

QoreChain's reinforcement learning consensus agent is branded **PRISM** (Policy-driven Reinforcement-learning for Intelligent State Machines). PRISM continuously adjusts consensus parameters to optimize network performance across multiple objectives. Rather than requiring governance votes for every parameter change, PRISM enables rapid adaptation while maintaining safety through circuit breakers and staged rollout modes.

PRISM is structured as two cooperating roles that share a single MLP policy. The **inline inference path** (described in Section 10.6) executes a forward pass on every block using the currently published policy weights, producing parameter adjustments directly inside the block lifecycle. The **PRISM Trainer** runs off-path, observing network metrics every 10 blocks, computing PPO gradient updates, and publishing refreshed policy weights back to the inline path through an on-chain `MsgUpdatePolicy` transaction, which emits a `policy_updated` event. The Trainer never executes consensus logic itself; it only trains and delivers weights. This split keeps the per-block

inference path deterministic and fully on-chain, while allowing training to run on a slower cadence without blocking block production. The network architecture is a multilayer perceptron (MLP) with 25 input dimensions (observations), two hidden layers of 256 units each, and 5 output dimensions (adjustable parameters). The total parameter count of 73,733 is small enough to fit entirely within blockchain state but large enough to capture complex interactions.

The observation vector includes 25 dimensions capturing network state: block time variance, transaction throughput, fee distribution, validator participation, bridge transaction volume, cross-VM traffic, reputation score distribution, slashing events, governance participation, consensus finality metrics, gas utilization, delegator rewards, inflation rate, RL agent mode, network latency, validator uptime distribution, MEV metrics, network partition signals, contract deployment rate, and other network health indicators.

The action vector outputs adjustments to 5 parameters: target block time (within 3–9 second range), maximum gas per block, base transaction fee, inflation rate multiplier, and slashing severity multiplier. The agent operates in multiple modes to balance innovation with safety: Shadow Mode (agent suggests but does not execute changes), Conservative Mode (agent executes with 50% dampening applied to all recommended adjustments), Autonomous Mode (agent executes full recommendations), and Paused Mode (no agent activity).

The reward signal for training combines five objectives with weights: throughput (30%), finality (25%), decentralization (20%), MEV minimization (15%), and failed transaction minimization (10%). This multi-objective design ensures the agent optimizes for platform quality rather than any single metric.

Table 15.9: On-chain RL agent parameters

Parameter	Value	Configurable
Network architecture	MLP ($25 \to 256 \to 256 \to 5$)	Governance (Tier 1)
Total parameters	73,733	Derived from architecture
Arithmetic precision	Fixed-point at 10^8 scale	No
Training algorithm	PPO (Proximal Policy Optimisation)	Governance (Tier 1)
Observation interval	10 blocks	Governance (Tier 2)
Observation vector dimension	25	Derived from state features
Action vector dimension	5	Derived from adjustable params
Circuit breaker window	50 blocks	Governance (Tier 2)
Circuit breaker threshold	0.5 (on-time ratio)	Governance (Tier 2)
Shadow mode epochs	50	Governance (Tier 2)
Conservative dampening factor	0.5	Governance (Tier 2)
Agent modes	Shadow, Conservative, Autonomous, Paused	Governance (Tier 1)

Code examples for interacting with the RL agent:

```
# Query current RL agent status and mode
qorechaind query rlconsensus agent-status --output json

# Example response:
# {
#   "mode": "conservative",
#   "parameters_adjusted_this_epoch": 3,
#   "recent_adjustments": [
#     {
#       "parameter": "target_block_time",
```

```
#       "old_value": 6,
#       "new_value": 6.1,
#       "block_height": 5240100
#     }
#   ],
#   "circuit_breaker_active": false,
#   "recommended_mode": "autonomous"
# }

# Query current observation vector
qorechaind query rlconsensus observation --output json

# Query reward signal components
qorechaind query rlconsensus reward-breakdown --output json

# Submit proposal to change RL agent mode (governance)
qorechaind tx gov submit-proposal rl-mode-change \
  --new-mode autonomous \
  --from validator \
  --output json

# Query agent performance metrics
qorechaind query rlconsensus metrics --lookback-epochs 100 --output json
```

15.3.6 QDRW Governance Voting

Quadratic Delegation with Reputation-Weighted (QDRW) governance voting, an optional tally extension disabled at genesis and activatable by governance proposal, combines three mechanisms to decentralize governance authority: quadratic voting (reducing plutocracy), delegation (enabling participation without expertise overhead), and reputation weighting (incentivizing long-term network stewardship). Until activation, governance voting uses the duration-weighted model (lock multipliers 1.0x to 2.0x).

Quadratic voting scales voting power as the square root of stake, ensuring that increasing stake gives diminishing additional voting power. This is a key innovation reducing dominance by large token holders. Combined with delegation, which allows token holders to assign their voting power to elected delegates, the system enables both direct participation and representative governance.

The reputation component adds a multiplier to voting power based on validator reputation scores. A validator with perfect reputation ($r = 1.0$) receives a 2.0x governance multiplier, while a validator with reputation 0.5 receives a 1.25x multiplier, and validators with reputation below 0.5 receive a 0.5x multiplier. This ensures that delegates with consistent track records of good behavior have disproportionate influence on governance decisions.

The xQORE (locked QOR) component further amplifies voting power through a governance multiplier. A delegator who locks QOR in the xQORE contract receives additional voting power, incentivizing long-term commitment. The xQORE multiplier is fixed at 2.0, meaning that 1 xQORE gives as much voting power as 2 QOR of delegated stake.

The voting power formula combines these components:

$$VP_i = \sqrt{s_i + 2x_i} \cdot Q(r_i), \qquad Q(r_i) = 0.5 + 1.5r_i \tag{15.12}$$

where s_i is staked QOR, x_i is xQORE held, and $Q(r_i)$ is the reputation multiplier function. The factor of 2 in $\sqrt{s_i + 2x_i}$ reflects the 2x governance multiplier for xQORE, effectively doubling the impact of locked stake in the square root calculation.

Table 15.10: QDRW governance parameters

Parameter	**Value**	**Configurable**
xQORE multiplier	2.0 (double weight)	Governance (Tier 1)
Reputation min multiplier	0.5	Governance (Tier 1)
Reputation max multiplier	2.0	Governance (Tier 1)
Quorum (Tier 1: Constitutional)	50%	Governance (Tier 1)
Quorum (Tier 2: Standard)	33.4%	Governance (Tier 2)
Quorum (Tier 3: Operational)	45%	Governance (Tier 1)
Approval (Tier 1)	66.7% (supermajority)	Governance (Tier 1)
Approval (Tier 2)	50% (simple majority)	Governance (Tier 2)
Approval (Tier 3)	66.7% (supermajority)	Governance (Tier 1)
Veto threshold (all tiers)	33.4%	Governance (Tier 1)
Voting period (Tier 1)	100,800 blocks (≈7 days)	Governance (Tier 1)
Voting period (Tier 2)	50,400 blocks (≈3.5 days)	Governance (Tier 2)
Voting period (Tier 3)	14,400 blocks (≈1 day)	Governance (Tier 1)
Execution delay (Tier 1)	28,800 blocks (≈2 days)	Governance (Tier 1)
Execution delay (Tier 2)	14,400 blocks (≈1 day)	Governance (Tier 2)
Execution delay (Tier 3)	0 (immediate)	No
Base deposit	10,000 QOR	Governance (Tier 2)
Deposit multiplier (Tier 1)	5x	Governance (Tier 1)
Deposit multiplier (Tier 3)	3x	Governance (Tier 1)

Code examples for calculating and querying governance voting power:

```
# Calculate voting power for a delegator
qorechaind query governance calc-voting-power \
  --delegator qor1abc...def \
  --output json

# Example response:
# {
#   "address": "qor1abc...def",
#   "staked_qor": 1000.0,
#   "xqore_locked": 500.0,
#   "reputation_score": 0.85,
#   "base_voting_power": 40.825,
#   "reputation_multiplier": 1.775,
#   "final_voting_power": 72.46,
#   "delegation_to": "qorvaloper1xyz...abc"
# }

# Query governance proposal with voting breakdown
```

```
qorechaind query gov proposal 123 --output json

# Submit a Tier 2 governance proposal (standard)
qorechaind tx gov submit-proposal param-change \
  --proposal-file proposal.json \
  --from delegator \
  --output json

# Vote on a proposal with breakdown by voting power
qorechaind tx gov vote 123 yes --from delegator --output json

# Query voting power leaderboard
qorechaind query governance voting-power-leaderboard \
  --limit 50 --output json
```

15.4 Virtual Machine Specifications

15.4.1 EVM (Ethereum Virtual Machine)

The Ethereum Virtual Machine (EVM) execution environment on QoreChain provides byte-code compatibility with Ethereum smart contracts, enabling developers to deploy existing Solidity contracts with minimal changes. QoreChain targets Ethereum Shanghai compatibility, supporting the latest instruction set and EIP-1559 dynamic fee model, while adding post-quantum cryptographic precompiles for enhanced security.

The EVM executes contract bytecode in a sandboxed environment with defined gas costs for each operation. Gas provides a mechanism to limit computation and prevent denial-of-service attacks through infinite loops or expensive operations. QoreChain uses the standard Ethereum gas schedule, ensuring that contracts behave identically on QoreChain and Ethereum in terms of computational costs and execution model.

QoreChain extends the standard EVM with four custom precompiles accessible at special addresses: `qor_pqc_verify` at `0x0800` for verifying ML-DSA-87 signatures, `qor_pqc_sign` at `0x0801` for generating signatures within contracts, `qor_cosmwasm_call` at `0x0810` for calling CosmWasm contracts from EVM code, and `qor_svm_call` at `0x0811` for executing SVM programs. These precompiles enable cross-VM interaction and quantum-safe cryptographic operations directly within Solidity contracts.

The maximum contract size limit of 24,576 bytes (EIP-170) prevents unbounded contract bytecode and protects the state database from excessive growth. Contracts exceeding this limit must either be split into multiple contracts or use assembly-based code compression techniques.

QoreChain provides 20 custom JSON-RPC methods beyond standard Ethereum JSON-RPC, enabling queries for PQC key status, bridge information, cross-VM message status, and RL agent metrics. These extensions maintain Ethereum JSON-RPC compatibility while providing deep QoreChain-specific functionality.

Table 15.11: EVM specifications

Parameter	Value
Solidity version support	0.8.x (via solc)
EVM compatibility	Ethereum Shanghai equivalent
Gas model	Standard Ethereum gas schedule
PQC precompile: `qor_pqc_verify`	Address: `0x0800`
PQC precompile: `qor_pqc_sign`	Address: `0x0801`
Cross-VM precompile: `qor_cosmwasm_call`	Address: `0x0810`
Cross-VM precompile: `qor_svm_call`	Address: `0x0811`
JSON-RPC compatibility	Ethereum JSON-RPC + 20 custom `qor_` methods
EIP-1559 fee model	Supported
Maximum contract size	24,576 bytes (EIP-170)

15.4.2 CosmWasm

CosmWasm is a WebAssembly-based smart contract platform designed for the Cosmos ecosystem, offering advantages over EVM-style languages: better resource isolation, language flexibility (Rust, AssemblyScript, etc.), and formal verification support. QoreChain integrates CosmWasm as a parallel execution environment, enabling Cosmos-native smart contracts while maintaining EVM compatibility through cross-VM bridges.

CosmWasm contracts are written primarily in Rust and compiled to WebAssembly (WASM), a portable binary instruction format. The WASM runtime is sandboxed from the host system and cannot access filesystem, network, or other system resources without explicit host function exports. This isolation model ensures that buggy or malicious contracts cannot compromise the node or affect other contracts.

The wasmd v0.60.5 runtime implements the CosmWasm standard with support for IBC (Inter-Blockchain Communication), enabling contracts to send and receive cross-chain messages natively. The 800 KB compressed bytecode size limit prevents storage bloat while supporting reasonably complex contracts. Memory limits of 32 MB per execution prevent runaway memory consumption during contract execution.

QoreChain provides the qorechain-pqc Rust crate for developers building CosmWasm contracts that require post-quantum cryptographic operations, enabling quantum-safe contract verification and signatures within the CosmWasm environment. Contracts can query blockchain state directly without the abstraction layers required in EVM, reducing gas costs for state-intensive operations.

Table 15.12: CosmWasm specifications

Parameter	Value
Runtime	wasmd v0.60.5
Language support	Rust (primary), AssemblyScript
Maximum contract bytecode	800 KB (compressed)
Memory limit per execution	32 MB
Gas metering	CosmWasm gas (mapped to SDK gas)
Supported operations	Upload, Instantiate, Execute, Migrate, Sudo
IBC contract support	IBC-enabled contracts (send/receive packets)
PQC SDK bindings	`qorechain-pqc` Rust crate

15.4.3 SVM (Solana Virtual Machine)

The Solana Virtual Machine (SVM) execution environment provides access to Solana-compatible smart contracts compiled to Berkeley Packet Filter (BPF) bytecode. This integration enables QoreChain to support the Solana developer ecosystem while maintaining distinct Cosmos-based consensus and state models. SVM programs are Turing-complete general-purpose executables, unlike the more limited EVM instruction set.

SVM programs are compiled from Rust source code to ELF binaries containing BPF bytecode. The BPF execution engine is JIT-compiled to native machine code on modern processors, providing performance comparable to native application code. Programs interact with the blockchain through read-only account state and stateful account modifications, following Solana's account model where data and code are separated.

The compute budget mechanism limits the resources consumed by each transaction. The default budget of 200,000 compute units is sufficient for typical transactions, while the maximum of 1,400,000 compute units allows computationally intensive operations (such as cryptographic verification or complex state queries) without blocking simpler transactions. Compute units are abstract units of work that account for CPU cycles, memory access, and disk I/O.

The rent model prevents account bloat by requiring accounts storing data to maintain a minimum balance proportional to their data size. This discourages unlimited state growth and incentivizes cleanup of unused accounts. Rent-exempt accounts with sufficient balance are exempted from rent payments indefinitely.

The address derivation scheme uses SHA-256 hashing of the deployer's public key and program bytecode, ensuring that two programs deployed by different parties are not confused even if bytecode is identical. This deterministic derivation enables cross-program calls to reliably target specific programs.

$$\text{compute\_cost}(i) = \sum_{j=1}^{n} \text{gas\_schedule}[op_j] \tag{15.13}$$

where op_j is the j-th operation executed in the program.

Table 15.13: SVM specifications

Parameter	Value
Execution engine	BPF (Berkeley Packet Filter)
Program format	ELF binary (compiled from Rust)
Compute budget (default)	200,000 compute units
Compute budget (maximum)	1,400,000 compute units
Account model	Solana-compatible (owner, data, lamports)
Rent model	Rent-exempt with minimum balance
Address derivation	SHA-256 of deployer + bytecode
JSON-RPC compatibility	Solana JSON-RPC subset (port 8899)
PQC key support	ML-DSA-87 keys via PQC account type

15.4.4 Cross-VM Communication

Cross-VM communication enables smart contracts in different execution environments (EVM, CosmWasm, SVM) to seamlessly interact within a single atomic transaction. Rather than treating the three VMs as isolated execution layers, QoreChain provides first-class primitives for cross-VM messages that execute within a single transaction context, maintaining ACID properties and atomic rollback on failure.

The cross-VM communication system uses a routing layer that translates calls between VM-specific calling conventions. An EVM contract calling a CosmWasm contract uses a special precompile at address 0x0810, which encodes the call and target address. The routing layer executes the CosmWasm contract synchronously, returns the result through standard return data mechanisms, and reverts the entire transaction if the CosmWasm contract fails.

Atomicity is a key property: if a three-way transaction spans EVM, CosmWasm, and SVM with modifications to state in each VM, all changes commit together or all rollback together. No partial states are visible to other transactions. This is achieved through the unified gas model and transaction sequencing: each transaction is processed atomically across all three VMs before the next transaction begins.

The maximum message payload of 64 KB provides sufficient capacity for complex contract calls including large data arrays or encoded structures while preventing unbounded message sizes that could cause denial-of-service. Gas accounting is unified: cross-VM calls consume gas in the calling VM and measured in the native unit, with conversions applied as needed.

Return data formats differ by VM: EVM uses ABI-encoded return data (following solidity function ABIs), CosmWasm uses JSON structures, and SVM uses Borsh binary serialization. The routing layer handles these conversions transparently, allowing the calling contract to receive results in its native format.

Table 15.14: Cross-VM messaging specifications

Parameter	Value
Supported directions	EVM ↔ CosmWasm ↔ SVM (all pairs)
Atomicity	Single-transaction atomic execution
Maximum message payload	64 KB
Gas accounting	Unified gas model across VMs
Return data format	ABI-encoded (EVM), JSON (CosmWasm), Borsh (SVM)

```
# Deploy a CosmWasm contract
qorechaind tx wasm store contract.wasm --from deployer --output json

# Deploy an SVM program
qorechaind tx svm deploy-program program.elf --from deployer --output json

# Query custom RPC methods
curl -X POST http://localhost:8545 \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","method":"qor_getPQCKeyStatus",
    "params":["qor1abc...def"],"id":1}'
```

15.5 Bridge and Interoperability Specifications

15.5.1 IBC Channel Configuration

Inter-Blockchain Communication (IBC) is a protocol for secure, reliable message passing between independent blockchains. QoreChain maintains 8 active IBC channels connecting to major Cosmos ecosystem chains, enabling trustless transfer of tokens and other assets between platforms. Each IBC channel establishes a dedicated communication path with confirmed proof of payment and settlement on both sides.

IBC channels rely on proofs submitted by relayers (off-chain actors) that attest to transaction finality on the remote chain. For Cosmos Hub and most Cosmos chains, finality is achieved within approximately 7 seconds (after the commit block is signed). For rollup-based chains like Arbitrum and Optimism, IBC integration waits for finality of the underlying Ethereum Layer 1, which requires 12 Ethereum blocks (approximately 3 minutes).

The IBC transfer module uses a lock-and-mint model: when transferring a token from Chain A to Chain B, the token is locked in an IBC escrow account on Chain A, and a corresponding wrapped token is minted on Chain B. The wrapped token can be transferred back through the reverse IBC channel to unlock the original token. This model ensures that the total supply across both chains remains constant.

Each channel uses connection-specific settings including packet timeout periods (typically 10 minutes), maximum packet sizes (typically several MB), and ordering guarantees (ordered or unordered). Ordered channels preserve message sequence, useful for transactions that depend on prior transactions. Unordered channels allow parallel processing and higher throughput but require higher-level protocols to ensure consistency.

Table 15.15: IBC channel specifications

Channel	Destination	Connection Type
channel-0	Cosmos Hub	IBC transfer
channel-1	Osmosis	IBC transfer
channel-2	Akash	IBC transfer
channel-3	Juno	IBC transfer
channel-4	Stargaze	IBC transfer
channel-5	Stride	IBC transfer
channel-6	Injective	IBC transfer
channel-7	Celestia	IBC transfer

Code examples for IBC operations:

```
# Query IBC channels
qorechaind query ibc channel channels --output json

# Transfer tokens via IBC to another chain
qorechaind tx ibc-transfer transfer \
  transfer \
  channel-0 \
  cosmos1xyz...abc \
```

```
  1000uqor \
  --from myaccount \
  --output json

# Query IBC channel statistics and packet data
qorechaind query ibc channel channel-client transfer channel-0 --output
    ↪ json

# Query in-flight IBC packets
qorechaind query ibc channel unreceived-packets transfer channel-0
    ↪ --output json

# Monitor IBC packet acknowledgments
qorechaind query ibc channel acks transfer channel-0 --output json
```

15.5.2 QCB Bridge Endpoints

The QoreChain Bridge (QCB) protocol connects 17 non-IBC-reachable chains, expanding interoperability beyond the Cosmos ecosystem. Unlike IBC which relies on light clients and on-chain proof validation, QCB uses a multi-attestation model where a set of designated bridge validators independently verify cross-chain transactions and attest to their validity. When a supermajority of bridge validators attest to a transaction, it is considered confirmed on the destination chain.

Bridge endpoints are configured for different chain types and require different confirmation depths due to distinct finality models. Bitcoin, with proof-of-work finality, requires 6 blocks (approximately 1 hour). Ethereum and most EVM chains require 12 blocks (approximately 3 minutes). Layer 2 solutions like Arbitrum and Optimism require 12 blocks on the underlying Layer 1 (Ethereum), which is a stronger finality guarantee than relying on L2 reorganization resistance.

The bridge supports heterogeneous token sets per chain. Major tokens like USDC, USDT, and wrapped native coins (WBTC, WETH, etc.) are supported across multiple chains. Smaller tokens are selectively enabled based on liquidity and bridge validator consensus. Users can only bridge tokens that are explicitly enabled for their source and destination chain pair.

The confirmation depth for each chain is tuned to ensure that the probability of a reorganization affecting the bridged transaction is below 10^{-6} (one in a million). This calibration is performed off-chain and encoded in QCB bridge configuration, updated periodically as network parameters change.

15.5.3 Bridge Security Parameters

Bridge security depends on three layers of defense: cryptographic attestation, circuit breaker anomaly detection, and AI-based fraud scoring. The attestation layer requires signatures from a supermajority of bridge validators using post-quantum ML-DSA-87 signatures, ensuring that attacks require compromising multiple independent entities and quantum computers cannot forge historical signatures.

The circuit breaker mechanism provides a circuit breaker that triggers when anomalous bridge activity is detected. Anomaly sensitivity is calibrated at 3.0 standard de-

Table 15.16: QCB bridge endpoint specifications

Chain	Confirmation Depth	Chain Type	Supported Tokens
Ethereum	12 blocks	evm	ETH, USDC, USDT, WBTC
Solana	32 slots	solana	SOL, USDC
BNB Smart Chain	15 blocks	evm	BNB, USDC, BUSD
Avalanche (C-Chain)	12 blocks	evm	AVAX, USDC
Polygon	128 blocks	evm	MATIC, USDC
Arbitrum	12 blocks (L1 finality)	evm	ETH, ARB, USDC
Optimism	12 blocks (L1 finality)	evm	ETH, OP, USDC
Base	12 blocks (L1 finality)	evm	ETH, USDC
TON	12 blocks	ton	TON
Tron	19 blocks	tron	TRX, USDT
Near	5 blocks	near	NEAR
Cardano	15 blocks	cardano	ADA
Polkadot	6 blocks	polkadot	DOT
Aptos	6 blocks	aptos_move	APT
Sui	3 checkpoints	sui_move	SUI
Bitcoin	6 blocks	utxo	BTC
Tezos	6 blocks	tezos	XTZ

viations, meaning that transactions with characteristics more than 3 sigma away from historical norms trigger investigation. The circuit breaker pauses new bridge operations for a cooldown period (50 blocks), allowing operators to investigate and respond to potential attacks.

The AI verification layer uses an ensemble of machine learning models to detect fraud patterns. The Mahalanobis distance is computed for each transaction based on 25 observed dimensions (transaction size, token type, sender reputation, bridge usage patterns, time-of-day, etc.). Transactions with Mahalanobis distance exceeding the anomaly threshold (3.0 sigma) are flagged. A secondary isolated forest model provides independent fraud scoring in the range [0, 1], with transactions scoring above 0.8 flagged as high-risk.

Rate limiting prevents burst attacks by capping the maximum total value that can be bridged in a single hour, per-chain. This parameter is governance-configurable and is typically set based on the bridge's target user base and transaction volume.

The fraud detection accuracy is designed to achieve an AUC (Area Under the ROC Curve) of at least 0.995, meaning the model can correctly distinguish between normal and fraudulent transactions 99.5% of the time at any classification threshold. This translates to false positive rates below 0.5% when targeting 99% true positive rate.

$$\text{Anomaly score} = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})} \tag{15.14}$$

where $\mathbf{x}$ is the transaction vector, $\boldsymbol{\mu}$ is the historical mean, and $\boldsymbol{\Sigma}$ is the covariance matrix estimated from historical transaction data.

Code examples for bridge security operations:

```
# Query bridge security parameters and status
qorechaind query bridge security-params --chain ethereum --output json

# Example response:
# {
```

Table 15.17: Bridge security parameters

Parameter	Value	Configurable
Attestation signature algorithm	ML-DSA-87	No
Minimum attestation threshold	Chain-specific (see above)	Governance (Tier 2)
Circuit breaker: anomaly sensitivity (κ)	3.0 standard deviations	Governance (Tier 2)
Circuit breaker: cooldown period	50 blocks	Governance (Tier 2)
Minimum active bridge validators per chain	Chain-specific	Governance (Tier 2)
Rate limiting: maximum hourly volume	Per-chain configurable	Governance (Tier 2)
AI verification: anomaly detection AUC	≥ 0.995 (designed target)	N/A
AI verification: fraud scoring threshold	0.8	Governance (Tier 2)

```
#   "chain": "ethereum",
#   "confirmation_depth": 12,
#   "attestation_threshold": "7/10",
#   "circuit_breaker_active": false,
#   "anomaly_sensitivity_sigma": 3.0,
#   "fraud_score_threshold": 0.8,
#   "rate_limit_hourly": "500000000000uqor"
# }

# Query recent bridge transactions and their security scores
qorechaind query bridge transactions \
  --chain ethereum \
  --limit 10 \
  --output json

# Check if a specific transaction passed security checks
qorechaind query bridge verify-transaction \
  --chain ethereum \
  --tx-hash 0x1234... \
  --output json

# Query bridge validator signatures for a transaction
qorechaind query bridge attestations \
  --txid 123 \
  --output json

# Monitor circuit breaker status
qorechaind query bridge circuit-breaker-status --output json
```

15.6 Multi-Layer Architecture Specifications

The multi-layer architecture enables horizontal scaling by allowing multiple specialized layer types to operate in parallel while maintaining security inheritance from the main chain. Each layer type serves different use cases: sidechains for application-specific chains, paychains for high-throughput payments, and rollups for general computation with different settlement and verification models.

The main chain provides the security anchor: all layer 2 solutions inherit QoreChain's

consensus security and post-quantum cryptographic guarantees. State anchoring creates periodic proofs that layer 2 state is consistent with the main chain, enabling exit mechanisms and cross-layer messaging.

The governance framework enables deployment of new layer instances through Tier 2 governance, allowing the community to add capacity without requiring protocol-level upgrades. Each layer type has different operational requirements: sidechains require independent validator sets, paychains can use a single operator, and rollups can be deployed by any user through smart contracts on the main chain.

15.6.1 Layer Types

Table 15.18: Multi-layer architecture parameters

Parameter	Sidechain	Paychain	HCS	Rollup (RDK)
Max active instances	Governance-set	50	Governance-set	100
Min validators/operators	3	1	3	1
Min stake	100 QOR	100 QOR	100 QOR	10,000 QOR
Target block time	Chain-specific	500 ms	Chain-specific	Profile-dependent
State anchoring	Periodic	Settlement interval	Commitment only	Per-batch
Main chain security	Inherited	Inherited	Commitment-level	Settlement proof

15.6.2 RDK Rollup Profiles

The Rollup Development Kit (RDK) enables developers to deploy rollups optimized for specific use cases. Four preset profiles provide batteries-included configurations: DeFi (zero-knowledge proofs with STARK for lowest latency consistent with security), Gaming (optimistic proofs for sub-second latency), NFT (zero-knowledge with STARK and dedicated sequencer for deterministic ordering), and Enterprise (based rollups for highest throughput with traditional settlement).

DeFi rollups prioritize decentralization and finality guarantees through zk-STARK settlement. Gaming rollups accept slightly higher settlement latency (sub-second finality on the rollup, settlement confirmed on main chain within 1-2 blocks) to achieve lower transaction latency. NFT rollups use dedicated sequencers and zk-STARK settlement to ensure deterministic ordering and atomic state transitions, critical for NFT marketplace operations.

Enterprise rollups use "based" sequencing (delegation to main chain validators) and STARK settlement, providing enterprise-grade throughput (designed for 10,000+ TPS) while inheriting the security of QoreChain's validator set. Fee models are customizable: DeFi and NFT typically use market-based pricing, while Gaming and Enterprise can subsidize user fees to encourage adoption.

15.6.3 RDK Common Parameters

All rollups share common parameters controlling deployment, security, and data availability. The maximum of 100 rollups prevents unbounded growth that could degrade main chain performance. The minimum stake of 10,000 QOR ($\sim$50,000 USD assuming typical token economics) creates a resource barrier preventing frivolous deployments while remaining accessible to serious projects.

Table 15.19: RDK preset profile specifications

Parameter	DeFi	Gaming	NFT	Enterprise
Settlement mode	ZK	Optimistic	ZK	Based
Proof type	STARK	Fraud proof	STARK	STARK
Sequencer mode	Shared	Shared	Dedicated	Dedicated
VM type	EVM	EVM	EVM	EVM
Target latency	2 s	<1 s	2 s	1 s
Bridge type	Canonical	Native	Native	Native
Fee model	Market	Subsidized	Market	Subsidized

The creation burn rate of 1% removes a portion of the stake from circulation at deployment time, creating deflation as the rollup ecosystem grows. The challenge window of 7 days for optimistic rollups provides sufficient time for fraud detection; over this period, any validator can submit a fraud proof if the rollup's proposed state is invalid.

Data availability (DA) is a critical component for rollup security. The maximum blob size of 2 MB and retention period of 30 days provide sufficient capacity for typical rollup operations while limiting node storage requirements. Rollup operators can choose DA backends: inline storage (on the main chain), Celestia (external DA service), or QoreChain DA (dedicated DA infrastructure operated by validators).

The maximum of 10 batches per block prevents main chain transaction spikes from rollup state roots. Settlement modes (optimistic, ZK, based, sovereign) enable different security/latency/cost tradeoffs; developers choose the mode best suited to their application.

Table 15.20: RDK common parameters

Parameter	Default	Configurable
Maximum rollups	100	Governance (Tier 2)
Minimum stake	10,000 QOR	Governance (Tier 2)
Creation burn rate	1% of stake	Governance (Tier 2)
Challenge window (optimistic)	7 days	Governance (Tier 2)
Maximum DA blob size	2 MB	Governance (Tier 2)
DA retention period	432,000 blocks (≈30 days)	Governance (Tier 2)
Max batches per block	10	Governance (Tier 2)
DA backends	Inline, Celestia, QoreChain DA	Per-rollup
Settlement modes	Optimistic, ZK, Based, Sovereign	Per-rollup
Sequencer modes	Shared, Dedicated, Based	Per-rollup

15.7 Tokenomics Parameters

QoreChain's tokenomics architecture balances scheduled emission into circulation (commonly described as inflation, though total supply remains fixed at 4.5 billion QOR) to incentivize early participation with deflationary mechanisms (burning) to maintain long-term value. The emission schedule follows a predetermined decay curve,

reducing annual emission from 17.5% in Year 1 to 2% in Year 5 and beyond. The total token supply is fixed at 4.5 billion QOR, ensuring scarcity and a predictable emission schedule.

The emission model is epoch-based, computing new token issuance in discrete units each epoch (approximately once per day, or every 100 blocks at 6-second block time). This batching approach reduces on-chain computation compared to per-block emission while providing reasonable precision for the emission schedule.

Fee distribution splits transaction fees across five recipients: validators (37%) receive direct compensation for block production and consensus participation, protocol burn (30%) creates deflation to offset some issuance, treasury (20%) funds governance-approved spending, delegators (10%) receive a share for participating in securing the network, and light nodes (3%) are compensated for supporting the light client network.

The burn mechanism is particularly important for long-term sustainability: as transaction volume increases, the burn channel absorbs tokens, naturally creating negative supply pressure. Over time, as inflation decreases below burn rate, the network can transition to net deflation without requiring explicit governance changes.

Table 15.21: Token supply and emission parameters

Parameter	**Value**	**Configurable**
Total supply	4,500,000,000 QOR (fixed)	No
Emission model	Epoch-based decay	Governance (Tier 1)
Year 1 emission rate	17.5%	Governance (Tier 1)
Year 2 emission rate	11.0%	Governance (Tier 1)
Year 3–4 emission rate	7.0%	Governance (Tier 1)
Year 5+ emission rate	2.0%	Governance (Tier 1)
Epoch length (emission)	100 blocks	Governance (Tier 2)

Table 15.22: Fee distribution parameters

Recipient	**Share**	**Configurable**
Validators	37%	Governance (Tier 2)
Protocol burn	30%	Governance (Tier 2)
Treasury	20%	Governance (Tier 2)
Stakers (delegators)	10%	Governance (Tier 2)
Light nodes	3%	Governance (Tier 2)
Total	**100%**	

15.7.1 Burn Engine Channels

The 10-channel burn engine creates multiple, independent mechanisms for removing tokens from circulating supply. Each channel targets a different burn source, allowing governance to fine-tune the overall deflation rate by adjusting individual channels. The burn engine is designed to create net deflation over time as the network matures and transaction volume increases.

The primary burn channel is transaction fees (30%), which automatically burns a fraction of every transaction fee collected. This mechanism scales with network activity: higher transaction volume directly increases burn rate, creating economic feedback loops that stabilize token price. The governance penalty channel burns deposits on failed governance proposals, creating economic incentive alignment and preventing spam proposals.

The slashing burn channel removes a portion of validator stakes that are slashed for misbehavior, amplifying the penalty beyond the slashed amount and creating stronger deterrence. The bridge fee channel burns a component of cross-chain transaction fees, reflecting the cost of maintaining secure bridge infrastructure. The spam deterrent channel creates a minimum transaction fee floor to prevent transaction flooding attacks.

The epoch excess channel burns any emission that exceeds the target, implementing an automatic rebalancing mechanism. If inflation rises above target (due to high validator reward claims or other factors), the excess is automatically burned. The manual burn channel allows governance to order direct token burns for market support or other purposes.

Contract callbacks enable smart contracts to burn tokens programmatically, useful for deflationary DeFi protocols or time-decay mechanisms. Cross-VM fee burning applies to cross-VM messages, ensuring that interoperability features contribute to deflationary pressure. Rollup creation fees burn 1% of the minimum stake required to create new rollups, creating persistent burn as the rollup ecosystem grows.

Table 15.23: 10-channel burn engine

Channel	Description
`tx_fee`	30% of all transaction fees
`governance_penalty`	Deposits burned on vetoed/failed-quorum proposals
`slashing_burn`	Portion of slashed validator stake
`bridge_fee`	Bridge transfer fee burn component
`spam_deterrent`	Minimum transaction fee floor
`epoch_excess`	Excess emission above target burned
`manual_burn`	Governance-initiated manual burns
`contract_callback`	Smart contract-initiated burns
`cross_vm_fee`	Cross-VM messaging fee burn
`rollup_create`	1% of rollup creation stake

15.7.2 xQORE Parameters

xQORE (locked QOR) is a governance-enhanced token created by locking QOR for a time period. The locking mechanism incentivizes long-term token holding by providing governance power amplification and the ability to earn governance rewards. The xQORE system implements progressive unlock, where the voting power multiplier decreases as the unlock deadline approaches, eventually reaching zero at full unlock.

The lock-to-mint ratio is 1:1, meaning locking 1 QOR creates exactly 1 xQORE. However, the xQORE governance multiplier of 2.0 means that 1 xQORE has the voting power of 2 QOR of delegated stake, creating a 2x governance premium for locked

holders. This premium incentivizes concentration of voting power among long-term believers in the protocol.

The exit penalty structure creates progressive incentives: immediate exits (less than 30 days) lose 50% of locked tokens to burn and rebase, 30-89 day exits lose 35%, 90-179 day exits lose 15%, and 180-day (full unlock) exits have 0% penalty. The penalties are redistributed to remaining xQORE holders through a PvP (proof-of-participation) rebase mechanism, rewarding users who maintain longer locks.

The full unlock duration of 180 days provides a six-month horizon for committed governance participation. This duration balances providing meaningful commitment signals with allowing sufficient flexibility for life changes or portfolio rebalancing. Governance votes can propose changing the full unlock duration through Tier 2 governance.

The xQORE system creates a bifurcated governance structure: casual token holders participate via standard delegation, while committed long-term holders receive governance power bonuses through xQORE. This separation allows protocols to adjust governance participation incentives independent of consensus incentives.

Table 15.24: xQORE governance-boosted staking parameters

Parameter	Value	Configurable
Lock-to-mint ratio	1:1 (QOR to xQORE)	No
Minimum lock duration	0 days (immediate exit with penalty)	No
Full unlock duration	180 days	Governance (Tier 2)
Exit penalty (<30 days)	50%	Governance (Tier 2)
Exit penalty (30–89 days)	35%	Governance (Tier 2)
Exit penalty (90–179 days)	15%	Governance (Tier 2)
Exit penalty (≥180 days)	0%	N/A
PvP rebase distribution	Penalties redistributed to remaining holders	No
Governance VP multiplier	2x (per unit xQORE vs staked QOR)	Governance (Tier 1)

15.8 QCAI Specifications

QCAI is QoreChain's integrated AI service providing three operational tiers: Fast (sub-second latency for simple operations), Balanced (5-second latency for standard operations), and Advanced (up to 30 seconds for complex analysis requiring deep formal verification). These tiers serve different use cases ranging from high-frequency trading to complex multi-contract audit scenarios.

The QCAI system operates as an off-path AI service, not part of the consensus-critical path. This architecture ensures that AI inference failures do not block blockchain operations while still providing cryptographic proofs of AI-generated recommendations. Clients can optionally verify QCAI recommendations against alternative models or traditional analysis before acting on them.

The AI models employ a three-layer defense against fraud and anomalies. Layer 1 uses statistical methods (Mahalanobis distance) to identify transactions with unusual statistical signatures. Layer 2 applies machine learning models (isolation forests, neural networks) trained on labeled historical fraud data to compute anomaly scores. Layer 3 analyzes cross-endpoint correlations over a 50-block window to detect coordinated attacks.

QCAI contract generation capabilities scale with service tier: the Fast tier generates simple token contracts (ERC-20 equivalent), the Balanced tier can generate multi-token DEXs and standard DeFi protocols, and the Advanced tier handles complex cross-VM interactions requiring formal verification hints. Contract audit depth similarly scales from surface-level pattern matching in Fast tier to deep formal verification in Advanced tier.

Table 15.25: QCAI service tier specifications

Property	**QCAI Fast**	**QCAI Balanced**	**QCAI Advanced**
Primary use case	Lightweight tasks	Standard operations	Deep analysis
Latency target	<1 s	<5 s	<30 s
Contract generation	Simple tokens	DeFi protocols	Complex cross-VM
Contract audit depth	Surface-level	Standard	Deep + formal hints
Anomaly detection	Basic threshold	Statistical + ML	Multi-model ensemble

Table 15.26: QCAI service and inference parameters

Parameter	**Value**	**Notes**
QCAI service communication	Dedicated service protocol	Non-consensus-critical
Anomaly detection: Mahalanobis threshold	3.0 σ	Layer 1 (statistical)
Fraud scoring: isolation forest threshold	0.8	Layer 2 (ML-based)
Cross-endpoint correlation window	50 blocks	Layer 3 (correlation)
Route optimisation: Pareto frontier	Multi-objective (fee, speed, security)	Real-time
LSTM load prediction horizon	Configurable (default: 100 blocks)	Predictive
Contract audit vulnerability classes	10 categories	See Chapter 13

Code examples for QCAI service interactions:

```
# Request AI service tier (Fast, Balanced, Advanced)
curl -s http://localhost:7070/qcai/generate-contract \
  -H "Content-Type: application/json" \
  -d '{
    "tier": "balanced",
    "contract_type": "dex",
    "tokens": ["USDC", "USDT", "QOR"],
    "features": ["swap", "liquidity_pools"]
  }' | jq

# Request smart contract audit via QCAI
qorechaind tx qcai audit-contract \
  --contract-address 0x1234... \
  --tier advanced \
  --from user \
  --output json

# Query anomaly detection status
curl -s http://localhost:7070/qcai/anomaly-score \
  -d '{"tx_hash":"0xabc..."}'

# Request route optimization via QCAI
```

```
curl -s http://localhost:7070/qcai/optimize-route \
  -H "Content-Type: application/json" \
  -d '{
    "source_chain": "ethereum",
    "dest_chain": "qorechain",
    "amount": "1000000000000000000",
    "token": "USDC",
    "objectives": {"fee": 0.4, "speed": 0.4, "security": 0.2}
  }' | jq
```

15.9 Light Node Specifications

Light nodes provide a scalable alternative to full validators for users who want to contribute to network security without running full node infrastructure. Two editions serve different purposes: SX (Server eXperience) light nodes run as persistent daemons on servers, while UX (User eXperience) nodes run on user devices (desktop, mobile) with graceful handling of offline periods and bandwidth constraints.

Light nodes perform state queries and transaction verification by submitting requests to full nodes, which provide proofs of correctness. The light node verifies these proofs without needing to download or store the full blockchain state, reducing storage requirements from gigabytes (full validator) to megabytes. This enables ecosystem participation on low-power devices like smartphones.

The registration fee of 1 QOR creates a spam barrier while remaining accessible to all participants. Light nodes provide regular heartbeats to the network (every 1,000 blocks, approximately 1.7 hours) proving continued operation. A grace period of 200 blocks allows temporary connectivity loss without jailing. Light nodes that maintain at least 80% uptime earn a 3% share of transaction fees, creating ongoing economic incentive for participation.

Light nodes use ML-DSA-87 keys derived through Argon2id key derivation, with local key encryption via AES-256-GCM. This provides quantum-safe key management even on user devices with limited security properties. The maximum of 10,000 light nodes prevents unbounded growth while supporting a large decentralized network.

$$\text{Light node reward} = \text{fee pool} \times 0.03 \times \frac{\text{uptime}}{\text{total light node uptime}} \times \mathbb{1}[\text{uptime} \geq 0.80] \tag{15.15}$$

Light node rewards are distributed proportionally by uptime, with nodes below the 80% threshold receiving zero reward for that epoch.

```
# Query light node parameters
qorechaind query lightnode params --output json

# Example response:
# {
#   "registration_fee": "1000000uqor",
#   "heartbeat_interval": 1000,
#   "heartbeat_grace_period": 200,
#   "min_delegated_stake": "100000000uqor",
```

Table 15.27: Light node parameters

Parameter	Value	Configurable
Editions	SX (Server eXperience), UX (User eXperience)	N/A
Registration fee	1 QOR	Governance (Tier 2)
Heartbeat interval	1,000 blocks (≈1.7 hours)	Governance (Tier 2)
Heartbeat grace period	200 blocks (≈20 minutes)	Governance (Tier 2)
Minimum delegated stake	100 QOR	Governance (Tier 2)
Reward share	3% of total transaction fees	Governance (Tier 2)
Minimum uptime for rewards	80%	Governance (Tier 2)
Maximum light nodes	10,000	Governance (Tier 2)
PQC keyring	ML-DSA-87 via native integration	No
Key derivation	Argon2id	No
Local key encryption	AES-256-GCM	No
Node types	`sx` (daemon), `ux` (web dashboard)	Per-node

```
#   "reward_share": "0.03",
#   "min_uptime_for_rewards": "0.80",
#   "max_light_nodes": 10000
# }

# Query light node network statistics
qorechaind query lightnode stats --output json
```

15.10 API and RPC Specifications

15.10.1 Custom JSON-RPC Methods

QoreChain extends the standard Ethereum JSON-RPC with 20 custom methods under the `qor_` namespace, providing deep introspection into quantum-safe cryptography, AI services, cross-chain operations, and consensus parameters. These methods enable developers and validators to query system state that is not available through standard Ethereum APIs while maintaining full backward compatibility with existing Ethereum tooling.

The custom methods follow Ethereum JSON-RPC 2.0 conventions with method names prefixed with `qor_` to avoid namespace collisions. Each method accepts standard JSON-RPC parameters and returns data encoded in JSON format. Error handling follows JSON-RPC error codes, with additional QoreChain-specific error codes for domain-specific failures.

Methods fall into several categories: cryptographic operations (getPQCKeyStatus, getHybridSignatureMode), AI and analytics (getAIStats, getSuggestRollupProfile), consensus state (getReputationScore, getPoolClassification, getRLAgentStatus), cross-chain operations (getCrossVMMessage, getBridgeStatus), multi-layer architecture (getLayerInfo, getRollupStatus), and tokenomics (getXQOREPosition, getBurnStats, getInflationRate).

The custom RPC interface provides a superset of capabilities beyond the standard Cosmos REST API and gRPC interfaces, specifically tailored for applications that need to monitor or control QoreChain-specific features. Since these methods are read-only queries (no state modification), they incur minimal computational overhead and can

be safely called at high frequency for monitoring applications.

Table 15.28: Custom `qor_` JSON-RPC methods

Method	Description
`qor_getPQCKeyStatus`	PQC key status for an address
`qor_getAIStats`	QCAI service statistics
`qor_getCrossVMMessage`	Cross-VM message status
`qor_getReputationScore`	Validator reputation score
`qor_getLayerInfo`	Multi-layer information
`qor_getBridgeStatus`	Bridge endpoint status
`qor_getBurnStats`	Burn engine statistics
`qor_getXQOREPosition`	xQORE position details
`qor_getInflationRate`	Current inflation rate
`qor_getTokenomicsOverview`	Tokenomics summary
`qor_getRLAgentStatus`	RL agent mode and metrics
`qor_getRLObservation`	Current RL observation vector
`qor_getRLReward`	Latest RL reward signal
`qor_getPoolClassification`	Validator pool assignment
`qor_getHybridSignatureMode`	Current PQC migration phase
`qor_getRollupStatus`	Rollup operational status
`qor_getListRollups`	List active rollups
`qor_getSettlementBatch`	Settlement batch details
`qor_getSuggestRollupProfile`	QCAI rollup profile suggestion
`qor_getDABlobStatus`	Data availability blob status

15.10.2 Standard Interfaces

QoreChain provides seven standard interfaces supporting multiple client ecosystems and protocols. The Ethereum JSON-RPC (port 8545) offers compatibility with Ethereum tools like Metamask and Ethers.js, enabling seamless integration of EVM contracts and tooling. The Solana JSON-RPC (port 8899) similarly provides Solana CLI and SDK compatibility for SVM programs.

The Cosmos ecosystem provides two complementary interfaces: REST API (port 1317) offers HTTP-based query and transaction submission with human-readable JSON responses, while gRPC (port 9090) provides high-performance binary protocol for backend-to-backend communication. gRPC is particularly suitable for validator nodes, relayers, and high-frequency applications due to superior performance.

The Consensus RPC (port 26657) provides access to consensus state, block data, and transaction information, supporting network monitoring, light client protocols, and validator coordination. This interface follows the Cosmos consensus node conventions. The P2P interface (port 26656) handles peer-to-peer gossip, block propagation, and transaction propagation; it is not intended for client access.

Prometheus metrics (port 26660) expose node and application metrics in Prometheus format, enabling integration with monitoring and alerting systems. Common metrics include block height, validator participation, transaction throughput, burn rate, and RL agent status.

Table 15.29: API interface specifications

Interface	Port (default)	Protocol
Ethereum JSON-RPC	8545	HTTP/WebSocket
Solana JSON-RPC	8899	HTTP
Cosmos REST API	1317	HTTP
Cosmos gRPC	9090	gRPC
Consensus RPC	26657	HTTP/WebSocket
P2P	26656	TCP
Prometheus metrics	26660	HTTP

```
# Query PQC key status via custom RPC
curl -s http://localhost:8545 \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","method":"qor_getPQCKeyStatus",
    "params":["qor1abc...def"],"id":1}' | jq

# Query bridge status via custom RPC
curl -s http://localhost:8545 \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","method":"qor_getBridgeStatus","params":[],"id":2}'
    ↪ | jq

# Query RL agent status
curl -s http://localhost:8545 \
  -H "Content-Type: application/json" \
  -d
    ↪ '{"jsonrpc":"2.0","method":"qor_getRLAgentStatus","params":[],"id":3}'
    ↪ | jq
```

15.11 Performance Design Targets

All performance figures in this section represent designed targets based on theoretical analysis and single-node profiling. Multi-node testnet benchmarks will be published as validation proceeds during Phase 5 (Section 14.7).

Table 15.30: Performance design targets

Metric	Design Target	Validation Status
Main chain TPS	5,000+	Designed, pending multi-node benchmark
Block time	6 seconds	Designed, pending multi-node validation
Finality	Sub-second (designed)	Pending multi-node validation
ML-DSA-87 sign overhead	~0.8 ms (x86)	Single-node profiled
ML-DSA-87 verify overhead	~0.3 ms (x86)	Single-node profiled
Batch verification speedup	2–3x over individual	Theoretical
State proof size (10^8 entries)	864 bytes	Calculated
Transaction envelope overhead (PQC vs ECDSA)	~7 KB additional	Calculated
Cross-VM message latency	Same-block execution	Designed
Paychain block time	500 ms (target)	Pending validation
Bridge transfer latency	Chain-specific + 1–2 QoreChain blocks	Designed

QoreChain targets 5,000+ transactions per second on the main chain, complemented by horizontal scaling through the multi-layer architecture. The 6-second target block time balances finality speed with consensus stability; shorter block times increase validator communication overhead, while longer times reduce throughput. Quantum-safe cryptography introduces overhead that is fully accounted for in the throughput targets.

State proofs provide succinct proofs that specific data exists in the blockchain state without requiring clients to download the entire state. A state proof for 100 million (10^8) entries compresses to 864 bytes using Merkle trees with modern hash functions, enabling light clients to verify state with minimal bandwidth. Cross-VM messages target same-block execution, providing atomic semantics where either all cross-VM operations within a block commit or all rollback.

Paychains (specialized sidechain for payments) target 500 millisecond block times, offering 2,000 TPS for payment-focused operations. Bridge transfer latency depends on the remote chain's finality characteristics; for Ethereum it is 12 blocks plus 1-2 QoreChain blocks for settlement (approximately 4-5 minutes).

15.11.1 PQC Overhead Analysis

The transition from ECDSA to ML-DSA-87 introduces measurable overhead in transaction size and signing time. The per-transaction overhead:

$$\Delta_{\text{size}} = (|pk_{\text{ML-DSA}}| + |\sigma_{\text{ML-DSA}}|) - (|pk_{\text{ECDSA}}| + |\sigma_{\text{ECDSA}}|) = (2{,}592 + 4{,}627) - (33 + 64) = 7{,}122 \text{ bytes} \tag{15.16}$$

For the designed TPS target of 5,000 transactions per second, the additional bandwidth requirement:

$$\Delta_{\text{bandwidth}} = 5{,}000 \times 7{,}122 = 35.6 \text{ MB/s} \tag{15.17}$$

This is within the capacity of modern data center networks and does not represent a binding constraint on throughput. The signing time overhead ($\sim$0.75 ms additional per transaction) is similarly non-binding, as signature generation is parallelisable and occurs client-side.

15.11.2 Throughput Scaling

Main chain throughput is complemented by horizontal scaling through the multi-layer architecture:

$$\text{TPS}_{\text{aggregate}} = \text{TPS}_{\text{main}} + \sum_{s \in \mathcal{S}} \text{TPS}_s + \sum_{p \in \mathcal{P}} \text{TPS}_p + \sum_{r \in \mathcal{R}} \text{TPS}_r \tag{15.18}$$

where $\mathcal{S}$, $\mathcal{P}$, and $\mathcal{R}$ are the sets of active sidechains, paychains, and rollups respectively. Each paychain adds up to 2,000 TPS (at 500 ms block time), and each rollup adds throughput according to its profile configuration. With 10 active paychains and 5 active rollups, the aggregate designed throughput target exceeds 25,000 TPS.

Chapter 16

Risk Analysis, Regulatory Framework, and Conclusions

This chapter presents a formal risk assessment of QoreChain's core subsystems, situates QoreChain within the global regulatory landscape, draws conclusions on the platform's readiness, and provides a comprehensive glossary of terms used throughout this whitepaper.

16.1 Risk Analysis

Risk management in a quantum-safe blockchain demands rigorous quantification across multiple threat domains. This section models risks in cryptography, consensus, cross-chain operations, smart contract execution, and performance, using probability-theoretic and decision-theoretic frameworks.

16.1.1 Cryptographic Risk

Post-Quantum Algorithm Maturity

QoreChain's post-quantum cryptographic suite (ML-DSA-87, ML-KEM-1024, SLH-DSA, SHAKE-256) is standardised by NIST under FIPS 203, 204, and 205. However, the relative youth of lattice-based and hash-based schemes compared to RSA/ECDSA introduces residual uncertainty.

Define the *cryptographic confidence function* $C(t)$ for an algorithm family as:

$$C(t) = 1 - e^{-\lambda t} \tag{16.1}$$

where t is the number of years since standardisation and λ is the scrutiny rate (publications, cryptanalytic attempts, and formal proofs per year). For lattice-based schemes:

$$\lambda_{\text{lattice}} \approx \frac{N_{\text{papers}}(t)}{t} \cdot \frac{1}{N_{\text{attacks}} - successful}(t) + 1 \tag{16.2}$$

As of 2026, the lattice problem (Module-LWE) underlying ML-DSA-87 and ML-KEM-1024 has withstood over 20 years of cryptanalytic scrutiny with no sub-exponential classical or quantum attack discovered. The best known quantum attack against Module-LWE with the parameters used in ML-KEM-1024 requires:

$$T_{\text{quantum}} = 2^{174} \text{ quantum gate operations (NIST Security Level 5)} \tag{16.3}$$

Harvest Now, Decrypt Later (HNDL) Threat Model

The HNDL attack model assumes an adversary records encrypted communications today for decryption once a cryptographically relevant quantum computer (CRQC) becomes available. The expected loss from HNDL exposure for a blockchain with sensitive long-lived state:

$$\mathbb{E}[\mathcal{L}_{\text{HNDL}}] = \sum_{i=1}^{N} V_i \cdot P(\text{CRQC} \leq T_i) \cdot (1 - \mathbb{1}_{\text{PQC},i}) \tag{16.4}$$

where V_i is the value of asset i, T_i is its security horizon, $P(\text{CRQC} \leq T_i)$ is the probability of a CRQC arriving within that horizon, and $\mathbb{1}_{\text{PQC},i}$ is an indicator function equal to 1 if the asset is protected by post-quantum cryptography. For QoreChain, $\mathbb{1}_{\text{PQC},i} = 1$ for all native transactions, so $\mathbb{E}[\mathcal{L}_{\text{HNDL}}] = 0$ for on-chain assets. The residual risk exists only for assets bridged from non-PQC chains, which retain the source chain's cryptographic exposure until withdrawn.

Algorithm Agility and Migration Risk

Should a breakthrough reduce the security of ML-DSA-87, QoreChain's algorithm agility framework enables migration. The migration cost function:

$$\mathcal{C}_{\text{migrate}} = N_{\text{accounts}} \cdot c_{\text{rekey}} + N_{\text{validators}} \cdot c_{\text{rotate}} + c_{\text{governance}} \tag{16.5}$$

where c_{rekey} is the per-account cost of generating new key pairs, c_{rotate} is the per-validator key rotation cost, and $c_{\text{governance}}$ is the governance overhead for approving the algorithm change. The dual-signature transition mechanism allows overlapping validity of old and new signatures during migration:

$$\text{Valid}(tx) = \text{Verify}_{\text{old}}(\sigma_{\text{old}}, tx) \vee \text{Verify}_{\text{new}}(\sigma_{\text{new}}, tx) \quad \text{during epoch } [e_{\text{start}}, e_{\text{end}}] \tag{16.6}$$

This eliminates the "flag day" risk of instantaneous protocol-wide migration.

```
# Query current PQC algorithm parameters
qorechaind query pqc params --output json

# Example response:
# {
#   "signing_algorithm": "ML-DSA-87",
#   "kem_algorithm": "ML-KEM-1024",
#   "hash_algorithm": "SHAKE-256",
#   "security_level": 5,
#   "migration_status": "stable",
#   "next_algorithm": null,
#   "agility_epoch_window": 1000
# }
```

```
# Submit algorithm migration proposal (governance)
qorechaind tx gov submit-proposal pqc-migration \
  --new-signing-algo "SLH-DSA-256f" \
  --transition-epochs 500 \
  --from myvalidator \
  --output json
```

16.1.2 Consensus Risk

Byzantine Fault Tolerance Bounds

QoreChain's Combined Proof of Stake (CPoS) consensus inherits the classical BFT safety guarantee requiring fewer than one-third of weighted stake to be Byzantine. Define the Byzantine fraction f:

$$f = \frac{\sum_{i \in \mathcal{B}} s_i}{\sum_{j \in \mathcal{V}} s_j} \tag{16.7}$$

where $\mathcal{B} \subset \mathcal{V}$ is the set of Byzantine validators and s_i is the effective stake of validator i. Safety holds when $f < 1/3$. The probability of a safety violation given n validators, each failing independently with probability p:

$$P(\text{safety violation}) = \sum_{k=\lceil n/3 \rceil}^{n} \binom{n}{k} p^k (1-p)^{n-k} \tag{16.8}$$

For $n = 150$ validators and $p = 0.01$ (1% individual failure rate):

$$P(\text{safety violation}) < \binom{150}{50} (0.01)^{50} (0.99)^{100} \approx 10^{-78} \tag{16.9}$$

This bound is astronomically small, confirming that BFT safety is not a practical concern under realistic assumptions.

Liveness Risk Under Network Partition

Under a network partition separating the validator set into two groups of relative sizes α and $1 - \alpha$, liveness halts if neither partition holds a supermajority ($> 2/3$). The liveness failure condition:

$$\text{Liveness fails} \iff \max(\alpha, 1-\alpha) < \frac{2}{3} \iff \frac{1}{3} < \alpha < \frac{2}{3} \tag{16.10}$$

QoreChain's geographic distribution requirement (no single jurisdiction hosting more than 33% of validator stake) is designed to minimise the probability of a partition that splits the validator set within this range. The expected partition recovery time follows an exponential distribution:

$$P(\text{recovery} \leq t) = 1 - e^{-\mu t} \tag{16.11}$$

where μ is the network recovery rate, estimated at $\mu \approx 12$ hr^{-1} for modern internet infrastructure (median recovery time of 5 minutes).

Reputation Manipulation

The reputation component of CPoS introduces a secondary attack surface. An adversary attempting to maximise reputation score r_i through strategic behaviour faces the constraint that reputation is computed over a rolling window of W epochs:

$$r_i = \frac{1}{W} \sum_{e=t-W+1}^{t} (\omega_{\text{up}} \cdot u_{i,e} + \omega_{\text{sign}} \cdot b_{i,e} + \omega_{\text{gov}} \cdot g_{i,e} - \omega_{\text{slash}} \cdot s_{i,e}) \tag{16.12}$$

where $u_{i,e}$ is uptime, $b_{i,e}$ is block-signing participation, $g_{i,e}$ is governance participation, and $s_{i,e}$ is slashing history in epoch e. Since all components are observable on-chain and slashing events are irreversible within the window, the cost of reputation manipulation is:

$$\mathcal{C}_{\text{rep-attack}} \geq W \cdot c_{\text{honest-operation}} + \omega_{\text{slash}} \cdot \mathbb{E}[\text{slash penalty}] \tag{16.13}$$

The rolling window ensures that an adversary cannot build reputation faster than an honest validator, making long-term Sybil reputation attacks economically irrational.

```
# Query validator reputation breakdown
qorechaind query staking validator-reputation \
  qorvaloper1abc...def \
  --output json

# Example response:
# {
#   "validator": "qorvaloper1abc...def",
#   "reputation_score": 0.94,
#   "components": {
#     "uptime": 0.99,
#     "block_signing": 0.97,
#     "governance": 0.85,
#     "slashing_penalty": 0.00
#   },
#   "window_epochs": 100,
#   "rank": 12
# }
```

16.1.3 Cross-Chain Bridge Risk

Bridge Security Model

Cross-chain bridges are historically among the highest-risk components in blockchain infrastructure. QoreChain Bridge (QCB) employs a multi-attestation model where m of n bridge validators must attest to a cross-chain event. The probability of a successful bridge compromise:

$$P(\text{bridge compromise}) = \sum_{k=m}^{n} \binom{n}{k} p_c^k (1 - p_c)^{n-k} \tag{16.14}$$

where p_c is the probability that an individual bridge validator is compromised. For a 7-of-11 threshold with $p_c = 0.05$:

$$P(\text{bridge compromise}) = \sum_{k=7}^{11} \binom{11}{k} (0.05)^k (0.95)^{11-k} \approx 2.2 \times 10^{-7} \tag{16.15}$$

Value-at-Risk Analysis

The bridge Value-at-Risk (VaR) at confidence level α is:

$$\text{VaR}_\alpha = F^{-1}(\alpha) \cdot \text{TVL}_{\text{bridge}} \tag{16.16}$$

where F^{-1} is the inverse CDF of the loss distribution. Under a compound Poisson model for bridge incidents:

$$\mathbb{E}[\text{annual loss}] = \lambda_{\text{incident}} \cdot \mathbb{E}[L] = \lambda_{\text{incident}} \cdot \bar{l} \cdot \text{TVL}_{\text{bridge}} \tag{16.17}$$

where $\lambda_{\text{incident}}$ is the expected number of incidents per year and $\bar{l}$ is the average loss fraction per incident. QoreChain mitigates this through rate limiting (maximum transfer size per epoch), time-locked withdrawals for large amounts, and multi-layer attestation across both IBC and QCB protocols.

Confirmation Depth and Finality Risk

For each connected chain j, the required confirmation depth d_j is calibrated to ensure the reorganisation probability is below a threshold ϵ:

$$P(\text{reorg} \geq d_j) = \left(\frac{q_j}{1 - q_j}\right)^{d_j} < \epsilon \tag{16.18}$$

where q_j is the fraction of hash rate (PoW chains) or adversarial stake (PoS chains) on chain j. Solving for the minimum confirmation depth:

$$d_j > \frac{\ln \epsilon}{\ln\left(\frac{q_j}{1-q_j}\right)} \tag{16.19}$$

For Bitcoin with $q = 0.25$ and $\epsilon = 10^{-6}$:

$$d_{\text{BTC}} > \frac{\ln(10^{-6})}{\ln(1/3)} \approx 12.6 \implies d_{\text{BTC}} = 13 \text{ blocks} \tag{16.20}$$

```
# Query bridge risk parameters for a specific chain
qorechaind query bridge risk-params --chain ethereum --output json

# Example response:
# {
#   "chain": "ethereum",
#   "confirmation_depth": 64,
#   "max_transfer_per_epoch": "1000000000000uqor",
#   "attestation_threshold": "7/11",
#   "time_lock_threshold": "500000000000uqor",
#   "time_lock_duration": "24h",
#   "reorganisation_probability": "1e-9"
# }
```

16.1.4 Smart Contract Risk

Formal Verification Coverage

Smart contract vulnerabilities remain a primary risk vector in blockchain ecosystems. QoreChain's triple-VM architecture introduces distinct risk profiles for each execution environment. The residual vulnerability density ρ after k audit passes:

$$\rho(k) = \rho_0 \cdot (1 - \eta)^k \tag{16.21}$$

where ρ_0 is the initial vulnerability density (defects per KLOC) and η is the detection efficiency per audit pass. For a three-pass audit process with $\eta = 0.6$ and $\rho_0 = 15$ defects/KLOC (industry average for smart contracts):

$$\rho(3) = 15 \cdot (1 - 0.6)^3 = 15 \cdot 0.064 = 0.96 \text{ defects/KLOC} \tag{16.22}$$

QoreChain Studio's QCAI-powered audit combines static analysis, dynamic testing, and AI pattern recognition. The combined detection efficiency:

$$\eta_{\text{combined}} = 1 - (1 - \eta_{\text{static}})(1 - \eta_{\text{dynamic}})(1 - \eta_{\text{AI}}) \tag{16.23}$$

With $\eta_{\text{static}} = 0.55$, $\eta_{\text{dynamic}} = 0.45$, and $\eta_{\text{AI}} = 0.40$:

$$\eta_{\text{combined}} = 1 - (0.45)(0.55)(0.60) = 1 - 0.1485 = 0.8515 \tag{16.24}$$

This yields a residual vulnerability density after a single combined pass of $\rho(1) = 15 \times 0.1485 = 2.23$ defects/KLOC, comparable to three sequential traditional audit passes.

Cross-VM Interaction Risk

Cross-VM calls introduce composability risk, where a vulnerability in one VM can propagate through cross-VM message passing. The cross-VM risk amplification factor:

$$\mathcal{R}_{\text{cross-VM}} = 1 + \sum_{i \neq j} \frac{|\text{calls}_{i \to j}|}{|\text{calls}_{\text{total}}|} \cdot \left(\frac{\rho_i + \rho_j}{2} \right) \tag{16.25}$$

where $|\text{calls}_{i \to j}|$ is the volume of cross-VM calls from VM i to VM j. QoreChain mitigates this through sandboxed execution contexts, gas metering at VM boundaries, and mandatory type checking on cross-VM message payloads.

```
# Query smart contract audit status
qorechaind query studio audit-report \
  --contract qor1contract...addr \
  --output json

# Example response:
# {
#   "contract": "qor1contract...addr",
#   "audit_passes": 2,
#   "static_analysis": {"issues_found": 3, "severity": "low"},
#   "dynamic_testing": {"issues_found": 1, "severity": "medium"},
#   "ai_pattern_check": {"issues_found": 0},
```

```
#   "residual_risk_score": 0.12,
#   "recommendation": "deploy_with_monitoring"
# }
```

16.1.5 Performance Risk

Throughput Degradation Model

As network load approaches designed capacity, transaction throughput may degrade according to an M/M/1 queuing model. The expected transaction latency:

$$\mathbb{E}[T] = \frac{1}{\mu - \lambda} \tag{16.26}$$

where μ is the designed processing rate (5,000 TPS target) and λ is the arrival rate. The system remains stable when $\rho_{\text{util}} = \lambda/\mu < 1$. At 80% utilisation ($\lambda = 4{,}000$ TPS):

$$\mathbb{E}[T] = \frac{1}{5{,}000 - 4{,}000} = 1 \text{ ms} \tag{16.27}$$

At 95% utilisation ($\lambda = 4{,}750$ TPS):

$$\mathbb{E}[T] = \frac{1}{5{,}000 - 4{,}750} = 4 \text{ ms} \tag{16.28}$$

The multi-layer architecture provides overflow capacity: transactions exceeding main chain capacity are routed to paychains or sidechains, maintaining effective throughput beyond the main chain's designed limit.

State Growth and Storage Risk

Blockchain state grows monotonically, creating long-term storage costs. The state growth rate:

$$\frac{d|\mathcal{S}|}{dt} = \text{TPS} \cdot \bar{s}_{\text{tx}} \cdot (1 - \gamma_{\text{prune}}) \tag{16.29}$$

where $\bar{s}_{\text{tx}}$ is the average state change per transaction and γ_{prune} is the state pruning efficiency. With PQC-enlarged transactions ($\bar{s}_{\text{tx}} \approx 8$ KB including ML-DSA-87 signatures), the annual state growth at designed capacity:

$$\Delta|\mathcal{S}|_{\text{annual}} = 5{,}000 \times 8 \times 10^3 \times 3.15 \times 10^7 \times (1 - \gamma_{\text{prune}}) \text{ bytes} \tag{16.30}$$

At $\gamma_{\text{prune}} = 0.7$ (archival nodes prune 70% of historical state):

$$\Delta|\mathcal{S}|_{\text{annual}} \approx 378 \text{ TB (full nodes)} \quad \text{or} \quad 113 \text{ TB (pruned nodes)} \tag{16.31}$$

Light nodes, by design, store only block headers and Merkle proofs, reducing storage to approximately 50 GB per year. These figures are within range of enterprise-grade storage infrastructure and decline proportionally if average network utilisation is below designed capacity.

Table 16.1: QoreChain Risk Assessment Matrix

Risk Category	Specific Risk	Likelihood	Impact	Residual Risk
Cryptographic	CRQC breaks ML-DSA-87 (10 yr)	Very Low	Critical	Low (agility)
Cryptographic	HNDL exposure (native assets)	None	N/A	None
Cryptographic	HNDL exposure (bridged assets)	Medium	High	Medium
Consensus	BFT safety violation	Negligible	Critical	Negligible
Consensus	Liveness halt (partition)	Low	High	Low (geo-diversity)
Consensus	Reputation manipulation	Low	Medium	Low (rolling window)
Bridge	Multi-validator compromise	Very Low	Critical	Low (threshold)
Bridge	Source chain reorganisation	Low	High	Low (confirmation depth)
Smart Contract	Undiscovered vulnerability	Medium	High	Medium (multi-audit)
Smart Contract	Cross-VM propagation	Low	High	Low (sandboxing)
Performance	Main chain congestion	Medium	Medium	Low (multi-layer)
Performance	State bloat	Medium	Medium	Low (pruning)

16.1.6 Risk Summary Matrix

16.2 Regulatory Framework

QoreChain operates within a carefully constructed regulatory framework that prioritises compliance, transparency, and jurisdictional clarity. This section outlines the Swiss regulatory foundation, the classification of the QOR token, and alignment with international regulatory developments.

16.2.1 Swiss Regulatory Foundation

The Swiss DLT Act

Switzerland's Federal Act on the Adaptation of Federal Law to Developments in Distributed Ledger Technology (the "DLT Act"), which entered into force in stages between 1 August 2021 and 1 August 2022, established Switzerland as one of the most legally clear jurisdictions for blockchain projects. The legislative framework reflects a deliberate policy choice to integrate distributed ledger technology into existing financial market regulation rather than marginalising it through prohibition or creating parallel regulatory silos. This approach positions Switzerland as a destination for compliant blockchain infrastructure development. The DLT Act's passage followed comprehensive consultations with the Swiss Financial Market Supervisory Authority (FINMA), the State Secretariat for International Finance, and industry stakeholders, resulting in a framework addressing both pre-existing legal gaps and novel considerations specific to distributed ledger technology.

Key provisions include:

- **DLT Securities (Art. 973d et seq. CO):** Recognition of ledger-based securities ("Registerwertrechte") enabling tokenised financial instruments with full legal validity. These instruments satisfy all attributes of traditional securities (ownership rights, alienability, fungibility) while being recorded on a distributed ledger, creating a novel legal form that extends beyond centralised registry models. This provision enabled secondary markets for DLT-issued securities without requiring establishment of new exchanges.

- **DLT Trading Facilities (Art. 73a FinMIA):** A new license category for platforms trading DLT securities, providing a regulated pathway for decentralised exchanges. This licensing framework distinguishes between fully decentralised protocols (which operate without a single licensee) and DLT trading facilities operated by identifiable entities, enabling regulatory oversight where concentration of control exists.

- **Segregation in Bankruptcy (Art. 242a DEBA):** Crypto-assets held in custody are segregated from the bankrupt estate, protecting token holders. This segregation principle mirrors traditional securities law but explicitly extends to DLT-recorded assets, creating bankruptcy-remote treatment for customer assets and eliminating the counterparty risk faced by holders on platforms lacking custodial clarity.

- **Technology Neutrality:** The DLT Act amends existing financial market laws rather than creating a separate regime, ensuring blockchain-based instruments are subject to the same substantive rules as traditional instruments. This prevents regulatory arbitrage while acknowledging that the technical implementation need not differ in regulatory treatment.

- **Amendment Scope:** The act modified six federal laws covering securities law, civil law incorporation of shares, bankruptcy law, and financial market infrastructure regulation, demonstrating comprehensive integration across the legal system rather than piecemeal adaptation.

The DLT Act's implementation via cantonal coordination and Federal Parliament approval represents a rigorous legislative process. QoreChain Association is incorporated in Rolle, canton of Vaud, Switzerland (CHE-484.963.998), positioning the project within this comprehensive regulatory framework and demonstrating commitment to operating in full accordance with Swiss law and international best practices.

FINMA Token Classification

The Swiss Financial Market Supervisory Authority (FINMA) classifies tokens into three categories based on their functional characteristics and intended use. This classification framework evolved from FINMA's 2018 guidance on Initial Coin Offerings and was formalised through subsequent guidance documents and individual assessment decisions. The framework serves as a reference point for determining which financial laws apply to a given token and associated issuance activities.

1. **Payment tokens** (Zahlungs-Token): Tokens intended as a means of payment for goods and services, or for money or value transfer. Under Swiss law, payment tokens that function as electronic money are classified as financial instruments under the Financial Market Infrastructure Act (FinfraG) if they are traded on an exchange or multilateral trading platform.

2. **Utility tokens** (Nutzungs-Token): Tokens that provide access to a digital application or service and are usable for this purpose at the time of issuance. The defining characteristic is that the functionality must be available at the time of token distribution; tokens promising future functionality do not qualify as utility

tokens. Utility token classification is sensitive to the scope of available functions, the control structure of the platform provider, and the presence of secondary market trading.

3. **Asset tokens** (Anlage-Token): Tokens representing assets such as claims against the issuer, membership rights, profit participation, or collateral rights. Asset tokens are treated as financial instruments under Swiss law if they embody rights analogous to equities, bonds, derivatives, or structured products. This includes debt securities, participatory interests, and derivatives on underlying assets.

These classifications are not mutually exclusive; hybrid tokens may fall into multiple categories simultaneously if they embody characteristics of more than one category. Hybrid classification is assessed according to the token's primary economic function and the rights it actually confers at issuance. FINMA's classification decisions are fact-dependent and based on a comprehensive analysis of the technical implementation, economic incentives, and governance structures surrounding each token.

QOR Token Classification

Following a formal submission inquiry (*Unterstellungsanfrage*) filed on 22 December 2025, FINMA issued its assessment on 22 January 2026. FINMA's assessment confirmed the following regarding the QOR token:

Utility Token Classification. The QOR token qualifies as a *Nutzungs-Token* (utility token) under Swiss financial market law. This classification is based on the token's six functional roles within the QoreChain ecosystem:

1. **Transaction execution:** QOR is used to initiate and pay for transactions on QoreChain.

2. **Network security (staking):** QOR is staked to participate in the QoreChain Consensus Algorithm (QCA), securing the network. Critically, QoreChain Association does not hold or act as custodian of staked QOR tokens.

3. **Rewards:** QOR serves as the reward mechanism for ecosystem participation.

4. **Governance:** QOR holders participate in on-chain governance, voting on network upgrades, protocol rules, working groups, and grant committees.

5. **Grants and development:** QOR funds initiatives supporting developers and projects building on QoreChain.

6. **Smart contract access:** QOR provides access to QoreChain Studio, including the AI Smart Contract Generator and AI Smart Contract Auditor.

All six functions are available at the time of token issuance and operate within the QoreChain ecosystem, satisfying the core criterion for utility token classification.

Not a Payment Token. FINMA's assessment explicitly noted that the QOR token cannot be used as a means of payment for goods or services on the AI Marketplace or elsewhere. Commercial payments between users for products or services on the AI Marketplace are not conducted in QOR. As long as this condition is maintained, the QOR token does not qualify as a payment token under Swiss law.

Not an Asset Token. The QOR token does not represent a claim against QoreChain Association, does not confer membership rights in a corporate sense, and does not promise future capital flows or profit distributions. Token holders do not receive a repayment claim against QoreChain Association as the token issuer. The staking mechanism is technically necessary for protocol functionality and does not, in itself, constitute an investment function. As long as governance participation does not enable token holders to direct economic benefits to themselves, the QOR token does not qualify as an asset token.

Regulatory Implications. Under the utility token classification:

- The QOR token issuance does not constitute acceptance of public deposits under the Banking Act (BankG), as no repayment obligation exists.
- No authorisation is required under the Financial Institutions Act (FINIG).
- As long as the AI Marketplace does not facilitate trading of securities or other financial instruments, no authorisation is required under the Financial Market Infrastructure Act (FinfraG).
- QoreChain Association remains responsible for compliance with the Financial Services Act (FIDLEG) regarding prospectus obligations (Art. 35 ff. FIDLEG) and notification duties to FINMA (Art. 38 Abs. 3 FINMAG).

The formal mathematical representation of token classification as a decision function:

$$\text{Class}(\tau) = \begin{cases} \text{Payment} & \text{if } \exists \text{ goods/services exchangeable for } \tau \\ \text{Utility} & \text{if } \tau \text{ grants access to digital use/service at issuance} \\ \text{Asset} & \text{if } \tau \text{ represents claims, membership, or capital flows} \\ \text{Hybrid} & \text{if multiple conditions hold} \end{cases} \tag{16.32}$$

For QOR, the classification resolves to:

$$\text{Class}(\text{QOR}) = \text{Utility} \quad \because \quad \begin{cases} \text{Payment test: } \neg\exists\, g \in \mathcal{G} : \text{Pay}(\text{QOR}, g) = \text{true} \\ \text{Utility test: } \forall\, f \in \mathcal{F}_{\text{QOR}} : \text{Available}(f, t_{\text{issuance}}) = \text{true} \\ \text{Asset test: } \neg\exists\, \text{claim}(\text{QOR} \to \text{Issuer}) \end{cases} \tag{16.33}$$

where $\mathcal{G}$ is the set of goods/services on the AI Marketplace, $\mathcal{F}_{\text{QOR}}$ is the set of QOR token functions, and t_{issuance} is the issuance date.

```
# Query token regulatory parameters
qorechaind query tokenomics regulatory-info --output json

# Example response:
# {
#   "token": "QOR",
#   "classification": "utility",
#   "jurisdiction": "Switzerland",
```

```
#   "registrar": "CHE-484.963.998",
#   "regulator_assessment_date": "2026-01-22",
#   "functions": [
#     "transaction_execution",
#     "network_security_staking",
#     "rewards",
#     "governance",
#     "grants_development",
#     "smart_contract_access"
#   ],
#   "payment_function": false,
#   "custodial_staking": false,
#   "total_supply": "4500000000",
#   "denom": "uqor"
# }
```

16.2.2 Anti-Money Laundering Compliance

Under the Swiss Anti-Money Laundering Act (GwG, Geldwäschereigesetz), financial intermediaries engaged in handling or accepting customer assets must affiliate with a recognised self-regulatory organisation (SRO) under Art. 14 Abs. 1 GwG. The AML regulatory framework, as amended by the Federal Money Laundering Control Board, requires enhanced due diligence for beneficial ownership identification, transaction monitoring, and reporting of suspicious activities to the Swiss Financial Intelligence Unit (FIU). QoreChain's compliance framework is designed to address AML obligations through multi-layered technical and organisational controls:

1. **Non-custodial architecture:** QoreChain Association does not hold, custody, or exercise control over user funds. Staking is performed through non-custodial QoreChain wallets where users retain exclusive cryptographic control of their private keys; the Association does not act as a custodian of staked tokens. This architecture eliminates the requirement for AML controls on held assets, as the Association has no fiduciary relationship with token holders regarding staked balances. Transaction validation occurs at the protocol layer without asset settlement through the Association.

2. **On-chain transparency:** All transactions are recorded on a public, immutable ledger with full auditability. This enables transaction tracing and forensic investigation when required by law enforcement or regulatory authorities. The deterministic nature of on-chain records supports compliance verification without reliance on institutional record-keeping subject to loss or alteration.

3. **Bridge compliance:** Cross-chain transfers through the QoreChain Bridge (QCB) are subject to rate limiting (configurable thresholds per address and time period), time-locked withdrawals for large amounts (enforced at the smart contract level), and pluggable compliance modules that can integrate with third-party KYC/AML provider APIs. Bridge transactions can be frozen pending compliance verification, and transaction metadata is recorded for audit purposes.

4. **KYC/AML Integration Framework:** Exchanges and platforms deploying on QoreChain can implement Know Your Customer (KYC) and AML controls at the application layer, feeding regulatory designations (e.g., wallet sanctioning status) into smart contracts that enforce compliance rules. The modular architecture enables jurisdictional variation without requiring protocol-level code changes.

5. **SRO affiliation:** Where applicable based on operational activities (e.g., if the Association operates a designated exchange or wallet service), QoreChain Association is committed to affiliating with a recognised SRO under Swiss law, such as the Crypto Valley Association or a financial market SRO approved by FINMA.

16.2.3 International Regulatory Alignment

European Union: Markets in Crypto-Assets (MiCA)

Regulation (EU) 2023/1114, the Markets in Crypto-Assets Regulation (MiCA), fully applicable from 30 December 2024, establishes a comprehensive framework for crypto-assets in the EU. QoreChain's alignment with MiCA:

Table 16.2: QoreChain Alignment with EU MiCA Regulation

MiCA Requirement	QoreChain Status	Notes
Whitepaper publication (Art. 6)	Compliant	This document
Notification to competent authority (Art. 7)	Planned pre-TGE	Per jurisdiction of offering
Marketing communications (Art. 7)	Designed for compliance	Fair, clear, not misleading
Right of withdrawal (Art. 13)	Under legal review	14-day withdrawal period
Not an ART or EMT	Confirmed	QOR is a utility crypto-asset
Environmental sustainability (Art. 66)	Proof of Stake	No energy-intensive mining

Under MiCA's classification framework, QOR falls under the residual category of "crypto-assets other than ARTs or EMTs" (Title II), as it is neither an asset-referenced token (ART) nor an electronic money token (EMT). This is consistent with FINMA's utility token classification.

United States: Evolving Framework

The United States regulatory landscape for crypto-assets remains fragmented across the Securities and Exchange Commission (SEC), Commodity Futures Trading Commission (CFTC), and Financial Crimes Enforcement Network (FinCEN). QoreChain's approach:

- **Howey Test analysis:** The QOR token's utility functions, availability at issuance, and absence of profit-sharing mechanisms are designed to minimise classification risk as a security under the Howey test.
- **FinCEN compliance:** QoreChain's non-custodial architecture reduces obligations under the Bank Secrecy Act, though evolving regulatory guidance is monitored continuously.
- **Geographic restrictions:** Token distribution may be restricted in jurisdictions where regulatory clarity is insufficient, with compliance enforced through smart contract-level access controls.

CNSA 2.0 Alignment

The U.S. National Security Agency's Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) mandates a transition to quantum-resistant algorithms for national security systems by 2035. QoreChain's PQC suite aligns with CNSA 2.0 requirements:

Table 16.3: QoreChain PQC Alignment with CNSA 2.0

Function	CNSA 2.0 Requirement	QoreChain
Digital signatures	ML-DSA-87 (FIPS 204)	ML-DSA-87
Key encapsulation	ML-KEM-1024 (FIPS 203)	ML-KEM-1024
Hash-based signatures	SLH-DSA (FIPS 205)	SLH-DSA (backup)
Hashing	SHA-3 / SHAKE	SHAKE-256
Target completion	2035	Deployed at genesis

QoreChain's deployment of CNSA 2.0-aligned algorithms at genesis, rather than as a future migration, positions the platform as compliant with the most stringent national security cryptographic requirements from day one.

16.2.4 Compliance Risk Quantification

Regulatory compliance in multi-jurisdictional blockchain operations necessitates quantitative risk assessment across legal, financial, and operational dimensions. The expected annual compliance cost $\mathcal{C}_{\text{compliance}}$ can be modelled as:

$$\mathcal{C}_{\text{compliance}} = \sum_{j \in \mathcal{J}} \left(c_{\text{legal},j} + c_{\text{audit},j} + c_{\text{reporting},j} + c_{\text{legal counsel},j} \right) \cdot \mathbb{1}_{\text{active},j} \tag{16.34}$$

where $\mathcal{J}$ is the set of jurisdictions in which QoreChain operates, $c_{\text{legal},j}$ represents retainer and consultation costs for legal advisors specialising in DLT regulation in jurisdiction j, $c_{\text{audit},j}$ represents independent audit and assessment costs for regulatory compliance verification, $c_{\text{reporting},j}$ represents the cost of regulatory reporting and notification to supervisory authorities, and $\mathbb{1}_{\text{active},j}$ indicates whether QoreChain is active in that jurisdiction. Based on preliminary assessments, Europe (Switzerland, EU) represents approximately 40% of baseline compliance cost, North America (US) 35%, Asia-Pacific 15%, and emerging markets 10%.

The regulatory risk exposure quantifies the expected loss from adverse regulatory actions:

$$\mathcal{R}_{\text{regulatory}} = \sum_{j \in \mathcal{J}} P(\text{adverse ruling}_j) \cdot \mathcal{L}_j + \mathcal{R}_{\text{classification variance}} \tag{16.35}$$

where $P(\text{adverse ruling}_j)$ is the probability of an unfavourable regulatory action in jurisdiction j and $\mathcal{L}_j$ is the associated loss. Classification variance risk $\mathcal{R}_{\text{classification variance}}$ accounts for the possibility that jurisdictions update their regulatory positions, requiring operational or structural changes. Switzerland's formal classification via FINMA significantly reduces $P(\text{adverse ruling}_{\text{CH}})$ to below 5%, and provides a credible precedent that reduces uncertainty in other EU jurisdictions subject to regulatory harmonisation via MiCA. The residual risk in the United States

($P(\text{adverse ruling}_{\text{US}})$ estimated at 15-25%) primarily reflects the fragmented regulatory structure and potential SEC reclassification of utility tokens as securities based on economic substance analysis.

16.3 Conclusions

QoreChain presents a comprehensive response to the three convergent challenges facing contemporary blockchain infrastructure: the quantum computing threat to classical cryptography, the absence of native artificial intelligence integration, and the fragmentation of cross-chain interoperability.

16.3.1 Cryptographic Readiness

The deployment of NIST-standardised post-quantum algorithms (ML-DSA-87, ML-KEM-1024, SLH-DSA, SHAKE-256) at FIPS Security Level 5 positions QoreChain ahead of the quantum threat timeline. The Harvest Now, Decrypt Later (HNDL) attack vector, which threatens every blockchain relying exclusively on ECDSA or EdDSA, is neutralised for all native QoreChain transactions recorded from genesis onward. The quantum security posture is further strengthened by (1) hybrid signing during early phases, combining both classical and post-quantum signatures to defend against potential PQC algorithm breaks; (2) key rotation ceremonies conducted with cryptographic integrity verification; and (3) continuous evaluation of emerging quantum-resistant schemes through academic partnerships. The algorithm agility framework ensures that should cryptanalytic advances reduce confidence in any deployed algorithm, migration can occur through governance-controlled transitions without protocol disruption. Historical transaction data remains secure under the original cryptographic assumptions; only forward security requires algorithm migration. This defence-in-depth approach to cryptographic resilience mirrors the approach taken by critical national infrastructure, acknowledging that long-term blockchain security must account for threats extending decades beyond protocol deployment.

16.3.2 Architectural Differentiation

The triple-VM architecture (EVM, CosmWasm, SVM) operating within a unified state environment eliminates the need for developers to choose between ecosystem access and execution environment preference. This unified approach contrasts with existing multi-chain ecosystems where EVM contracts and non-EVM programs operate in isolated silos with limited composability. QoreChain's VM integration layer enables (1) direct calls between EVM contracts and CosmWasm/SVM programs with shared state visibility; (2) atomic cross-VM transactions that succeed or fail as a unit; and (3) a single fee mechanism that accounts for computation across all VMs. Combined with 25 direct bridge connections and over 120 chains reachable via IBC, QoreChain is designed to serve as a universal interoperability hub rather than an isolated execution layer. This positioning reduces fragmentation across the decentralised application ecosystem, enabling developers to build once and reach multiple blockchain user bases.

The multi-layer scaling architecture (main chain, sidechains, paychains, and rollups) provides horizontal throughput expansion beyond the main chain's designed 5,000+

TPS target, with aggregate network throughput targeting 25,000+ TPS across all scaling layers. The Rapid Deployment Kit (RDK) with preset profiles (DeFi, Gaming, NFT, Enterprise) lowers the barrier for application-specific chain deployment by providing pre-configured protocol parameters, validator set bootstrapping templates, and governance frameworks tailored to application requirements. This democratisation of chain deployment enables ecosystem participants to launch purpose-built execution environments without extensive cryptographic or protocol engineering expertise.

16.3.3 AI-Native Design

QCAI's three-tier architecture (Fast, Balanced, Advanced) provides intelligence services at every layer of the protocol stack, from transaction routing optimisation and predictive resource allocation to smart contract generation and security auditing in QoreChain Studio. The Fast tier enables real-time transaction classification and network congestion prediction; the Balanced tier provides multi-objective optimisation of validator set composition and stake distribution; the Advanced tier implements formal verification guidance and vulnerability detection in smart contracts. This stratification allows light clients and validators with limited computational resources to access intelligence services appropriate to their deployment constraints.

The reinforcement learning-based consensus parameter adaptation represents a departure from static protocol design toward systems that improve with operational experience. Consensus parameters such as block time, validator commission percentages, and gas costs are continuously optimised against reward signals derived from network health metrics (finality latency, crosschain bridge efficiency, application throughput). This adaptation mechanism enables QoreChain to respond automatically to changing network conditions without requiring governance votes for parameter updates, accelerating the protocol's ability to improve as transaction patterns emerge from the application ecosystem. The learning process is auditable and reversible, with governance retaining the ability to freeze parameter adaptation or specify constraints on permissible ranges.

16.3.4 Economic and Governance Sustainability

The QOR token's fixed supply of 4,500,000,000 tokens, announced at genesis with no future issuance (scarcity is guaranteed by consensus rules, not policy), combined with the multi-channel fee distribution model (37% validators, 30% burned, 20% treasury, 10% stakers, 3% light nodes), creates aligned incentives across all network participants. This fee structure is designed such that no single class of participants can sustain profitability if they act against protocol security; validator profitability depends on network health, treasury funds are deployed toward protocol development benefiting all holders, and token burn creates deflationary pressure that benefits long-term holders. The multi-channel distribution prevents concentration of economic power in any single participant group.

The Quadratic Delegation with Reputation Weighting (QDRW) governance extension (disabled at genesis, activatable by governance vote), with its mathematically bounded resistance to plutocratic capture (voting power increases sublinearly with stake) and flash-loan manipulation (reputation weight is updated at block finality, not on the same block as delegation), provides a governance framework designed for long-

term decentralisation. The system's security properties have been formally analysed in game-theoretic terms; the cost to acquire 67% of voting power (required for governance hijacking) grows quadratically with stake acquired, making sustained attacks economically prohibitive. Additionally, reputation weighting biases governance toward validators who have demonstrated long-term commitment to network security, reducing the influence of short-term speculators.

16.3.5 Regulatory Clarity

QoreChain's incorporation in Switzerland under the DLT Act (CHE-484.963.998), combined with FINMA's formal utility token classification of the QOR token (assessment issued January 22, 2026), provides a regulatory foundation that many blockchain projects lack. The FINMA classification covers all six functional uses of the QOR token and explicitly confirms that the token is not a payment token or asset token under Swiss law, eliminating regulatory ambiguity around token issuance. This clarity reduces the compliance uncertainty that plagues blockchain projects operating without formal regulatory guidance.

Alignment with EU Markets in Crypto-Assets (MiCA) positions QoreChain as a compliant crypto-asset infrastructure within the EU regulatory perimeter. The CNSA 2.0 cryptographic alignment further strengthens QoreChain's standing with U.S. national security agencies and critical infrastructure operators who may adopt blockchain systems only if they meet quantum-resistant cryptography mandates. The combination of Swiss legal incorporation, formal FINMA token classification, and cryptographic alignment with U.S. national security requirements creates a regulatory posture that is simultaneously protective of the project and attractive to institutional and government adoption.

16.3.6 Path Forward

With testnet validation ongoing (chain ID: `qorechain-diana`) and mainnet targeted for Q2 2026, QoreChain's development trajectory follows a rigorous validation methodology. Near-term milestones include (1) completion of independent security audits by tier-one blockchain security firms; (2) achievement of 5,000+ TPS sustained throughput on multi-node testnet with Byzantine-resilient adversarial conditions; (3) validation of post-quantum cryptographic performance with cryptographic side-channel analysis; (4) deployment and stress-testing of 10+ sidechain instances in production-like configurations; and (5) formal verification of critical consensus properties using machine-checked theorem provers.

The performance claims throughout this whitepaper are qualified with their validation status (designed, theoretical, single-node profiled, or pending multi-node validation), reflecting a commitment to scientific honesty. As testnet benchmarks mature and external audits complete, these claims will be updated with empirical evidence. Mainnet launch will occur only upon satisfaction of security and performance criteria, not according to a predetermined schedule.

The convergence of quantum-safe cryptography, AI-native intelligence, and universal interoperability in a single Layer-1 platform represents QoreChain's core thesis: that the next generation of blockchain infrastructure must be designed from first principles to address threats and opportunities that incremental upgrades to existing platforms

cannot adequately capture. The project invites developers, researchers, validators, and users to participate in translating this vision into a platform that advances the state of distributed systems technology and serves the evolving needs of decentralised applications.

16.4 Glossary

Term	Definition
AML	Anti-Money Laundering. Legal and regulatory framework to prevent money laundering and terrorist financing.
BFT	Byzantine Fault Tolerance. The ability of a distributed system to reach consensus even when up to one-third of participants act maliciously.
BPF	Berkeley Packet Filter. The bytecode format used by the Solana Virtual Machine for program execution, adopted by QoreChain's SVM layer.
CBDC	Central Bank Digital Currency. A digital form of central bank money designed for wholesale or retail use.
CNSA 2.0	Commercial National Security Algorithm Suite 2.0. NSA's mandate for quantum-resistant algorithms in national security systems by 2035.
Cosmos SDK	The open-source framework from which QoreChain evolved, providing the foundational modules for building application-specific blockchains.
CosmWasm	A WebAssembly-based smart contract platform integrated as one of QoreChain's three virtual machines.
CPoS	Combined Proof of Stake. QoreChain's consensus mechanism combining Reputation-Proof-of-Stake (RPoS), Delegated Proof-of-Stake (DPoS), and standard Proof-of-Stake (PoS) with weights 0.40, 0.35, and 0.25 respectively.
CRQC	Cryptographically Relevant Quantum Computer. A quantum computer capable of breaking current public-key cryptography (estimated at 4,000+ logical qubits for RSA-2048).
DLT	Distributed Ledger Technology. A digital system for recording, sharing, and synchronising data across multiple sites without a central administrator.

Term	Definition
DLT Act	Switzerland's Federal Act on the Adaptation of Federal Law to Developments in Distributed Ledger Technology, effective 2021–2022.
ECDSA	Elliptic Curve Digital Signature Algorithm. The classical signature scheme used by Bitcoin and Ethereum, vulnerable to quantum attack.
EVM	Ethereum Virtual Machine. The execution environment for Solidity smart contracts, natively supported by QoreChain.
FDV	Fully Diluted Valuation. The total market capitalisation if all tokens were in circulation (FDV = total supply × price).
FIDLEG	Swiss Financial Services Act (*Finanzdienstleistungsgesetz*). Regulates the provision of financial services and the offering of financial instruments.
FINIG	Swiss Financial Institutions Act (*Finanzinstitutsgesetz*). Governs the authorisation and supervision of financial institutions.
FinfraG	Swiss Financial Market Infrastructure Act (*Finanzmarktinfrastrukturgesetz*). Regulates financial market infrastructures including exchanges and trading systems.
FINMA	Swiss Financial Market Supervisory Authority (*Eidgenössische Finanzmarktaufsicht*). Switzerland's integrated financial markets regulator.
FINMAG	Swiss Financial Market Supervision Act (*Finanzmarktaufsichtsgesetz*). The foundational law governing FINMA's authority and operations.
FIPS 203	Federal Information Processing Standard for ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism).
FIPS 204	Federal Information Processing Standard for ML-DSA (Module-Lattice-Based Digital Signature Algorithm).
FIPS 205	Federal Information Processing Standard for SLH-DSA (Stateless Hash-Based Digital Signature Algorithm).
GwG	Swiss Anti-Money Laundering Act (*Geldwäschereigesetz*). Governs due diligence and reporting obligations for financial intermediaries.
HCS	Healthcare Chain System. A QoreChain sidechain configuration for healthcare data management with privacy-preserving computation.

Term	Definition
HNDL	Harvest Now, Decrypt Later. An attack strategy where encrypted data is captured today for decryption once quantum computers become available.
IBC	Inter-Blockchain Communication. A protocol for authenticated message passing between blockchains, used by QoreChain for cross-chain connectivity.
IoT	Internet of Things. Network of physical devices with embedded sensors, software, and connectivity. QoreChain supports lightweight IoT wallets for device-level blockchain participation.
KYC	Know Your Customer. Identity verification procedures required by financial regulations.
MiCA	Markets in Crypto-Assets Regulation (EU 2023/1114). The EU's comprehensive regulatory framework for crypto-assets, fully applicable from December 2024.
ML-DSA-87	Module-Lattice-Based Digital Signature Algorithm at NIST Security Level 5 (FIPS 204). QoreChain's primary digital signature algorithm.
ML-KEM-1024	Module-Lattice-Based Key-Encapsulation Mechanism at NIST Security Level 5 (FIPS 203). QoreChain's primary key exchange algorithm.
Module-LWE	Module Learning With Errors. The computational hardness assumption underlying ML-DSA and ML-KEM, believed resistant to both classical and quantum attacks.
One-Hop-Swap	QoreChain's cross-chain swap mechanism enabling direct asset exchange through QoreChain as a universal routing hub, without wrapped tokens.
PQC	Post-Quantum Cryptography. Cryptographic algorithms designed to resist attacks by both classical and quantum computers.
QCA	QoreChain Consensus Algorithm. The consensus mechanism implementing CPoS with AI-optimised validator selection and dynamic parameter adaptation.
QCAI	QoreChain AI Services. The AI-native intelligence layer with three tiers: Fast (low-latency inference), Balanced (general-purpose), and Advanced (complex reasoning).
QCB	QoreChain Bridge. The cross-chain bridge protocol connecting QoreChain to 17 external blockchain endpoints with multi-attestation security.

Term	Definition
QDRW	Quadratic Delegation with Reputation Weighting. Optional governance tally extension, disabled at genesis and activatable by governance proposal, with voting power formula $\mathrm{VP} = \sqrt{\text{staked} + 2 \cdot x_{\mathrm{QORE}}} \cdot R(r)$. At genesis, voting power is duration-weighted (1.0x-2.0x lock multipliers).
QOR	The native token of QoreChain, classified as a utility token under Swiss financial market law. Denominated in uqor (1 QOR = 10^6 uqor).
RDK	Rapid Deployment Kit. QoreChain's toolkit for deploying application-specific chains with preset profiles (DeFi, Gaming, NFT, Enterprise).
RPoS	Reputation-Proof-of-Stake. The reputation-weighted component of CPoS, contributing 40% of composite validator selection weight.
SHAKE-256	Extendable-output function from the SHA-3 family, used by QoreChain for state commitments, Merkle trees, and address derivation.
SLH-DSA	Stateless Hash-Based Digital Signature Algorithm (FIPS 205). QoreChain's backup signature scheme providing hash-based security independent of lattice assumptions.
SRO	Self-Regulatory Organisation. Under Swiss law (Art. 14 GwG), an organisation recognised by FINMA to supervise compliance of financial intermediaries with AML obligations.
SVM	Solana Virtual Machine. A BPF-based execution environment for high-performance smart contracts, integrated as QoreChain's third VM.
TGE	Token Generation Event. The initial distribution event for the QOR token.
TPS	Transactions Per Second. The throughput metric for blockchain performance. QoreChain's main chain is designed for 5,000+ TPS.
TVL	Total Value Locked. The aggregate value of assets deposited in a blockchain's DeFi protocols and bridges.
VaR	Value-at-Risk. A statistical measure of the potential loss in value of a portfolio over a defined period at a given confidence level.

Term	Definition
xQORE	Governance-boosted staking token. Created by committing QOR to an extended lock period, providing enhanced governance weight: duration-based multipliers of 1.0x to 2.0x at genesis, and a 2x weighting inside the QDRW formula if that extension is activated.