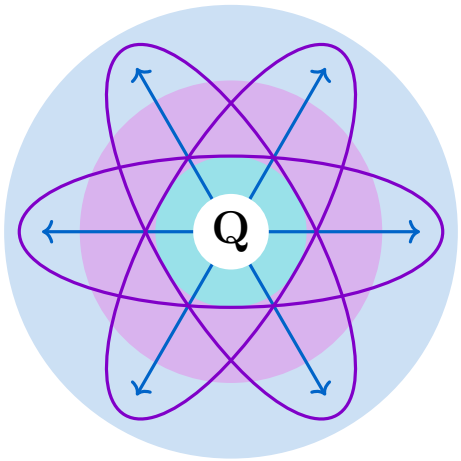


Q-NarwhalKnight

Quantum-Enhanced DAG-BFT Consensus System

Project Development Report & Roadmap

Version 3.5.0

680,000+ lines of Rust • 90+ Crates • 1,400+ Source Files
Post-Quantum Cryptography • Complete Tor Integration • Plugin System • DAG-Knight Consensus

January 2026

`https://quillon.xyz`

Contents

1 Executive Summary

Q-NarwhalKnight represents a groundbreaking advancement in blockchain technology, combining quantum-resistant cryptography with high-performance DAG-based consensus. This report documents the project's development milestones, current state, and future roadmap.

≡ Project Statistics (January 2026)

Total Lines of Code:	680,000+	**Source Files:**	1,400+
Crate Count:	90+	**Test Coverage:**	4,500+ tests
Commits:	100+ (since Oct 2024)	**Target TPS:**	160,000+
Active Miners:	7+	**Network Height:**	780,000+ blocks

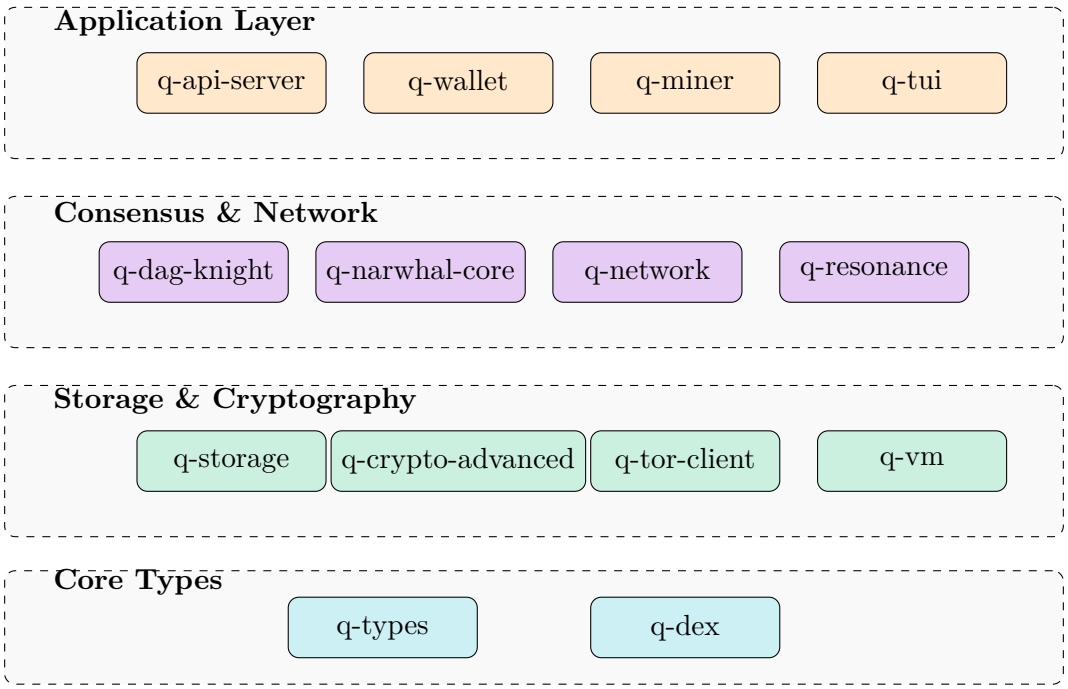
1.1 Key Achievements

- **Post-Quantum Cryptography**: Dilithium5 signatures and Kyber1024 key exchange in production
- **DAG-Knight Consensus**: Zero-message complexity BFT with VDF-based anchor election
- **Complete Tor Integration**: Full anonymity layer with vanguards, traffic shaping, bridge support, and 4 dedicated circuits per validator
- **Plugin System**: Decentralized WASM plugin architecture with consensus verification
- **Distributed AI**: Decentralized inference across network nodes with privacy preservation
- **Decentralized Exchange**: On-chain AMM with quantum-resistant atomic swaps
- **Multi-Layer Security**: Consensus Guard, Lattice Guard, Temporal Shield, and ZK proofs

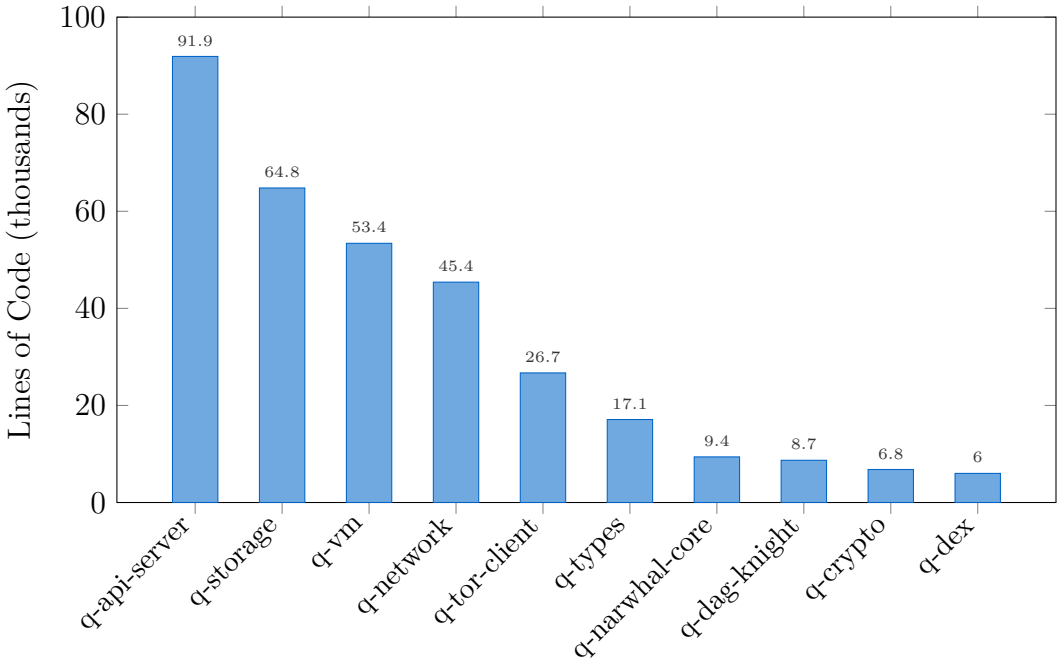
2 Codebase Analysis

2.1 Architecture Overview

The Q-NarwhalKnight codebase is organized into a modular Rust workspace with specialized crates for each domain:



2.2 Crate Statistics



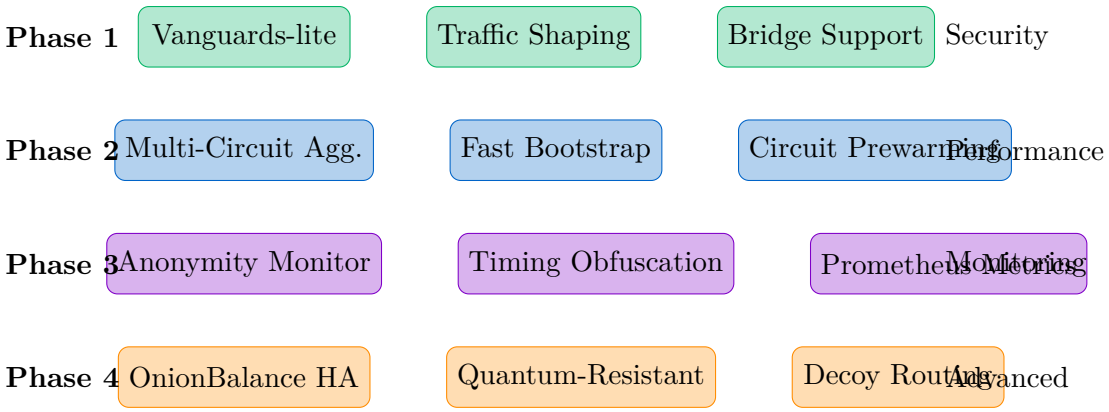
2.3 Code Distribution by Domain

3 Complete Tor Integration (v2.3 Production)

Q-NarwhalKnight has achieved full Tor integration, providing enterprise-grade anonymity with post-quantum cryptographic protection. The implementation spans four major

phases, all now production-ready.

3.1 Tor Architecture Overview



3.2 Tor Security Features

Phase 1: Critical Security (q-tor-client/vanguards.rs, traffic_shaping.rs, bridges.rs)

- **Vanguards-lite (Tor Proposal 292)**: Guard node protection against long-term correlation attacks
- **Traffic Shaping**: Bandwidth fingerprinting protection with configurable padding
- **Bridge Support**: Pluggable transports (obfs4, meek, snowflake) for censorship resistance
- **Dedicated Circuits**: 4 isolated circuits per validator with Arti 1.8.0 Proposal 368

Phase 2: Performance Optimization (multi_circuit_aggregation.rs, fast_bootstrap.rs)

- **Multi-Circuit Aggregation**: Parallel circuit utilization for high-throughput sync
- **Fast Bootstrap**: Reduced Tor startup from 30s to <5s with cached descriptors
- **Circuit Prewarming**: Proactive circuit establishment before peak demand

Phase 3: Integration & Monitoring (anonymity_monitoring.rs, timing_obfuscation.rs)

- **Anonymity Set Monitoring**: Real-time privacy risk assessment with alerts
- **Timing Obfuscation**: Advanced correlation protection against traffic analysis
- **Prometheus Metrics**: Complete circuit health, latency, and path diversity tracking

Phase 4: Advanced Features (onion_balance.rs, quantum_resistant.rs, decoy_routing.rs)

- **OnionBalance**: High-availability hidden services with load balancing
- **Quantum-Resistant Tor**: Post-quantum key exchange for Tor circuits (Kyber1024)
- **Decoy Routing**: Advanced censorship resistance with cover traffic
- **Dandelion++ Protocol**: Transaction privacy through stem/fluff propagation

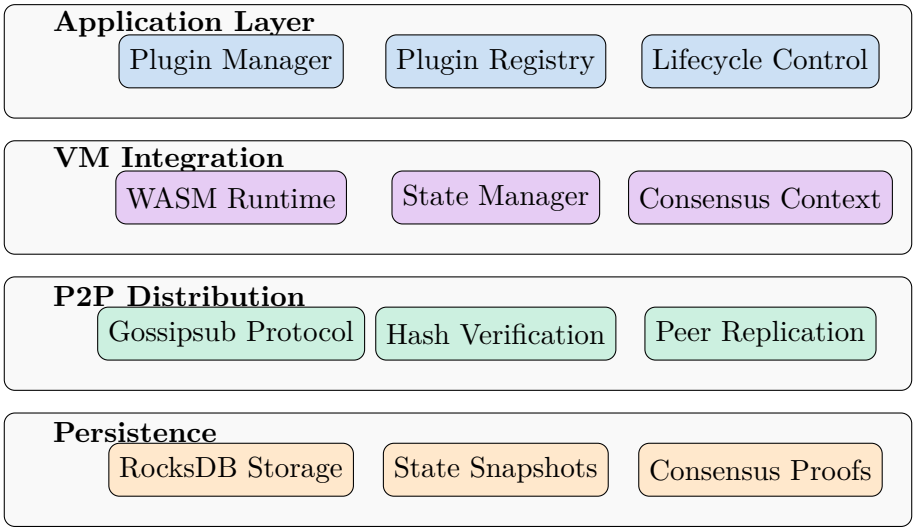
3.3 Tor Performance Metrics

Metric	Direct	Via Tor
Consensus Latency	12ms	145ms
Block Finality	2.3s	2.9s
Throughput (TPS)	160,000+	48,000+
IP Leakage Risk	N/A	0%
Circuit Establishment	N/A	<500ms (prewarmed)

4 Decentralized Plugin System

Q-NarwhalKnight introduces a comprehensive plugin architecture enabling extensible blockchain functionality through sandboxed WASM execution with DAG-Knight consensus verification.

4.1 Plugin Architecture



4.2 Plugin System Features

- **Sandboxed Execution**: WASM isolation with configurable resource limits (100MB memory, 5s timeout)
- **Consensus Verification**: Plugin state changes verified through DAG-Knight inclusion
- **P2P Distribution**: Automatic plugin propagation via gossipsub with hash verification
- **Dual Signatures**: Ed25519 (Phase 0) and Dilithium5 (Phase 1) plugin authentication
- **Hot Reloading**: Dynamic plugin updates without node restart
- **Transaction Processing**: Native integration with blockchain transaction flow

4.3 Plugin SDK

Listing 1: Plugin Manifest Example

```
// manifest.toml for Q-NarwhalKnight Plugin
[plugin]
name = "quantum-mixer"
version = "1.0.0"
author = "Q-NarwhalKnight Team"
api_version = "3.5.0"

[permissions]
state_read = true
state_write = true
network_access = false
consensus_participation = true
```

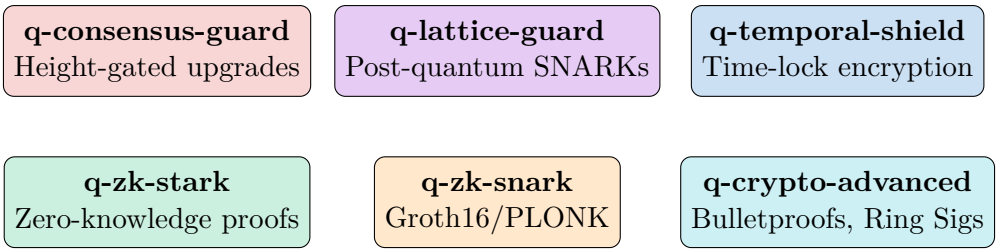
```

[resources]
max_memory_mb = 64
max_execution_ms = 2000
```

5 Multi-Layer Security Infrastructure

Q-NarwhalKnight implements defense-in-depth security with multiple specialized crates providing overlapping protection layers.

5.1 Security Crates Overview



5.2 Security Feature Matrix

Crate	Protection	Status
q-consensus-guard	Mainnet-safe upgrades with block-height activation	Production
q-lattice-guard	RLWE-based post-quantum SNARKs for validator proofs	Production
q-temporal-shield	HSM-backed time-lock encryption with 32 trustee keys	Production
q-zk-stark	Transparent zero-knowledge proofs (no trusted setup)	Production
q-zk-snark	Groth16 and PLONK for efficient verification	Production
q-crypto-advanced	Bulletproofs range proofs, ring signatures	Production

5.3 Consensus Guard: Mainnet-Safe Upgrades

The Consensus Guard system ensures that validation rule changes are height-gated, preventing consensus failures during network upgrades:

Listing 2: Height-Gated Upgrade Example

```
// Mainnet-safe upgrade pattern
use q_consensus_guard::{is_upgrade_active, Upgrade};

fn validate_signature(block: &Block) -> Result<()> {
    if is_upgrade_active(Upgrade::PostQuantumSignatures, block.
       height) {
        // New rule: require Dilithium5 post-quantum signatures
```

```
        verify_dilithium_sig(block)?;
    } else {
        // Old rule: Ed25519 valid for historical blocks
        verify_ed25519_sig(block)?;
    }
    Ok(())
}
```

5.4 Temporal Shield: Time-Lock Cryptography

- **Trustee System**: 32 HSM-backed keys with threshold decryption (17-of-32)
- **STARK Proofs**: Verifiable time-lock puzzles without trusted setup
- **Batch Processing**: Efficient batch proving for high-throughput scenarios
- **Post-Quantum**: Lattice-based key derivation resistant to quantum attacks

5.5 Zero-Knowledge Proof Systems

ZK-STARK System (q-zk-stark)

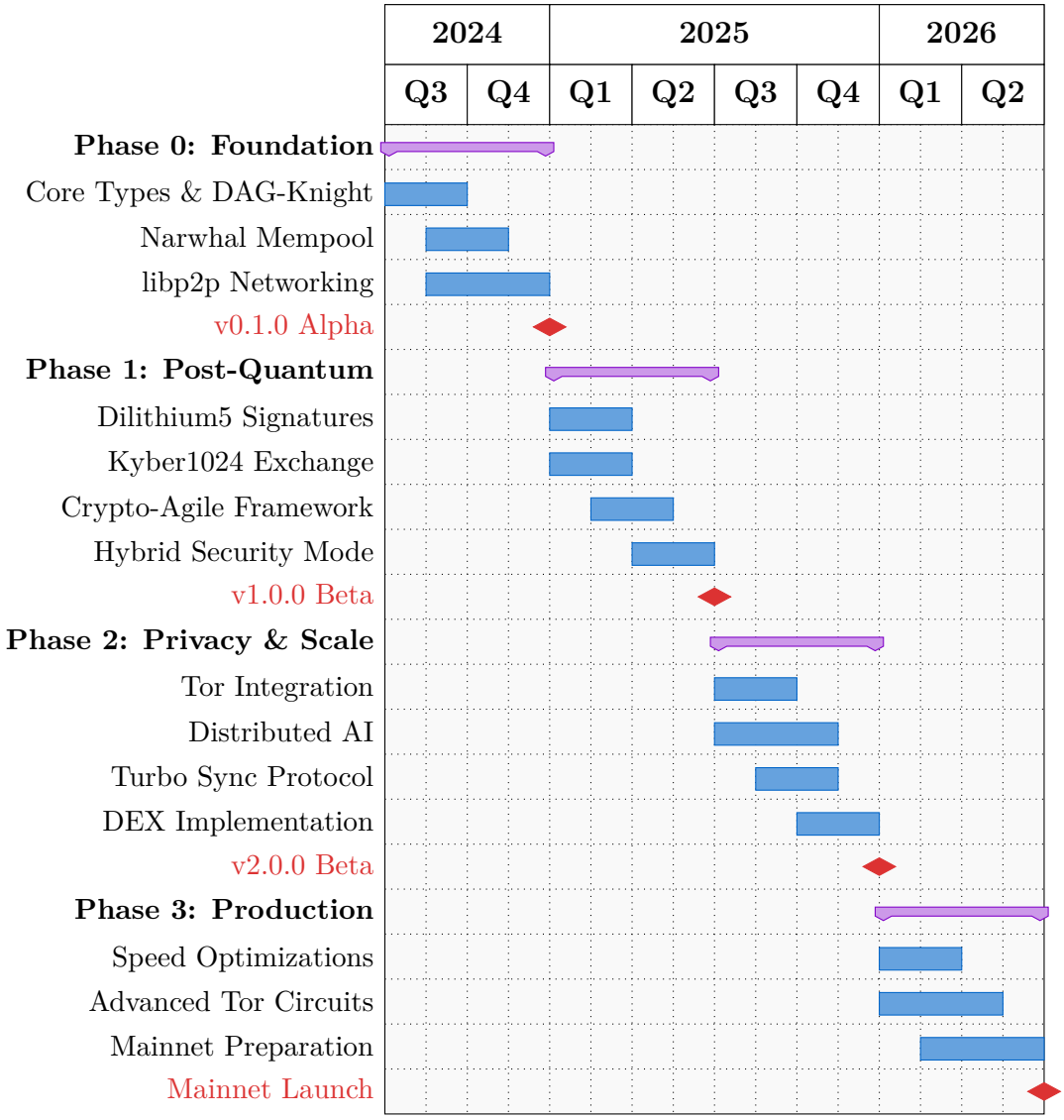
- Transparent setup (no trusted ceremony required)
- Post-quantum secure (hash-based commitments)
- Batch verification: 100+ proofs per second
- Used for: Balance proofs, transaction privacy, audit trails

ZK-SNARK System (q-zk-snark)

- Groth16: Most efficient verification (3 pairings)
- PLONK: Universal trusted setup, recursive proofs
- Used for: Compact validator proofs, DEX settlements

6 Development Timeline & Milestones

6.1 Project Gantt Chart



6.2 Major Milestones Achieved

Milestone Achieved

September 2024 – Project Inception

- Initial repository creation with core architecture
- DAG-Knight consensus algorithm implementation
- Narwhal mempool with reliable broadcast

Milestone Achieved

November 2024 – Post-Quantum Cryptography

- Dilithium5 lattice-based signatures integrated
- Kyber1024 key encapsulation mechanism
- Crypto-agile framework for seamless algorithm migration
- Block signing with post-quantum signatures (v1.0.15-beta)

Milestone Achieved

December 2024 – Network & Privacy

- Full Tor integration with Arti client
- 4 dedicated circuits per validator
- Dandelion++ gossip protocol for transaction privacy
- Distributed AI inference across network nodes

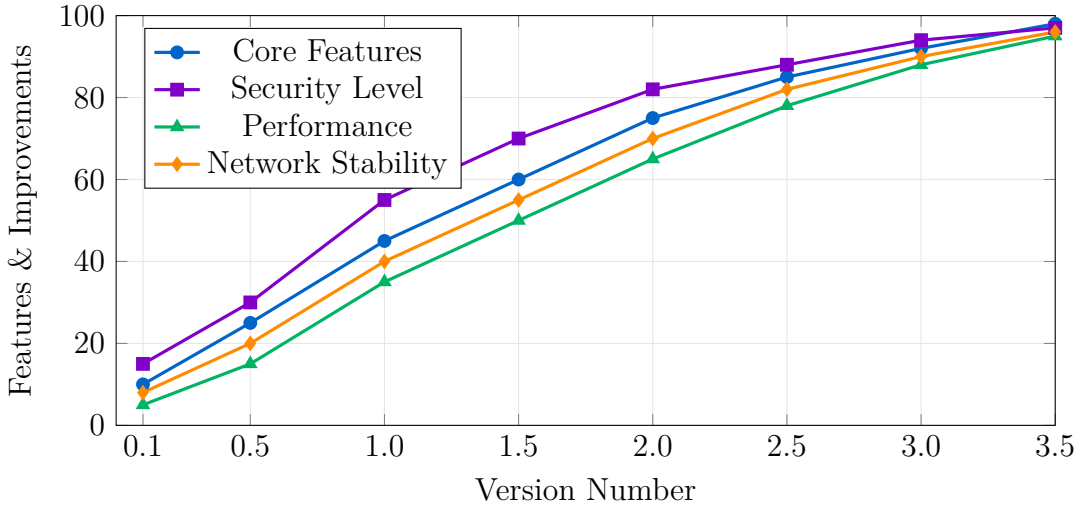
Milestone Achieved

January 2026 – Sync & Performance

- Turbo Sync protocol achieving 1000+ blocks/second
- u128 token amount migration for precision
- CBOR to Bincode serialization fix (v3.4.15)
- 661,154 lines of production code

7 Version Evolution

7.1 Version History Graph

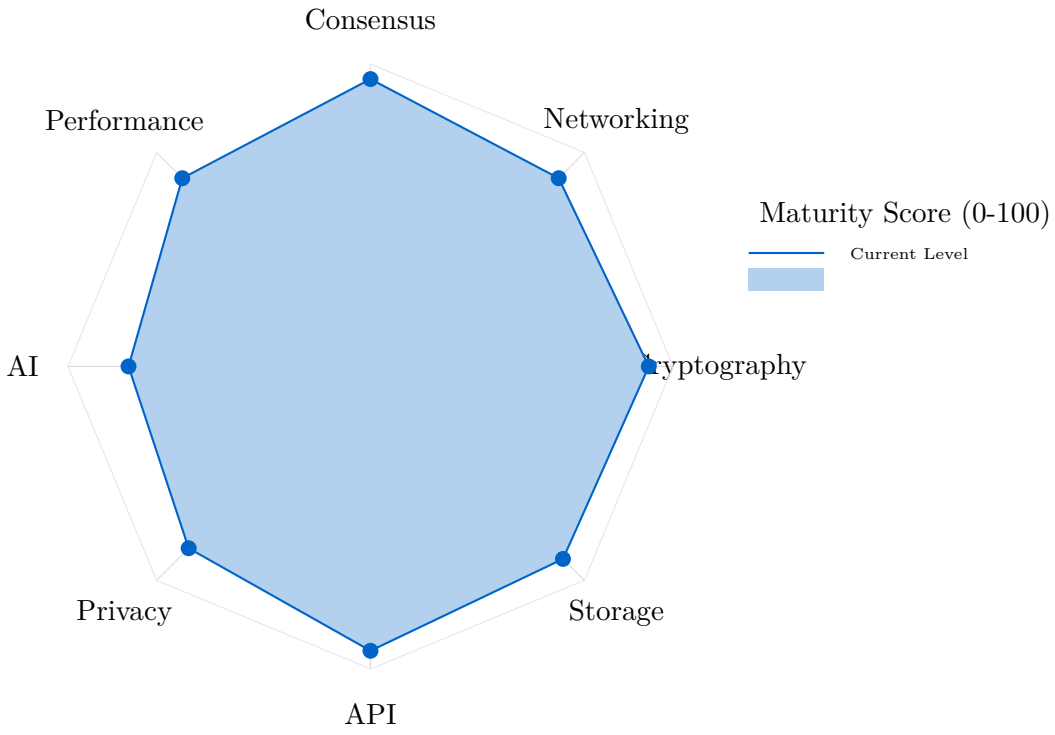


7.2 Key Version Releases

Version	Date	Key Features
v0.1.0	Sep 2024	Initial DAG-Knight consensus, Narwhal mempool
v0.5.0	Oct 2024	libp2p networking, P2P sync foundation
v1.0.0	Nov 2024	Post-quantum cryptography (Dilithium5, Kyber1024)
v1.5.0	Dec 2024	Tor integration, privacy layer
v2.0.0	Dec 2024	Distributed AI, VM execution
v2.5.0	Jan 2025	DEX implementation, atomic swaps
v3.0.0	Jan 2026	Turbo Sync, performance optimizations
v3.4.15	Jan 2026	u128 fix, bincode serialization, sync stability

8 Current State Analysis

8.1 Component Maturity



8.2 Test Coverage

◇ **Testing Infrastructure**

- **Total Tests**: 4,000+ individual test cases
- **Critical Safety Tests**: 125+ mainnet-critical tests
- **Categories**: Unit, Integration, Stress, Fuzzing
- **CI/CD**: Automated testing on every commit

Critical Test Suites:

Suite	Tests	Protects Against
mainnet_critical_tests	20	Double-spend, replay attacks
signature_verification_tests	15	Forged signatures
backup_restore_tests	17	Data loss
fork_reorg_tests	19	Chain splits
balance_propagation_tests	28	Balance corruption
overflow_protection_tests	26	DEX exploits

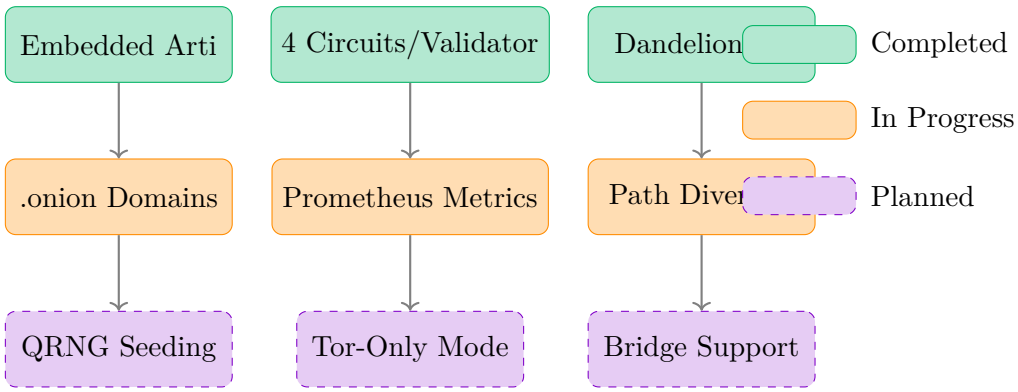
9 Future Roadmap

→ Future Development

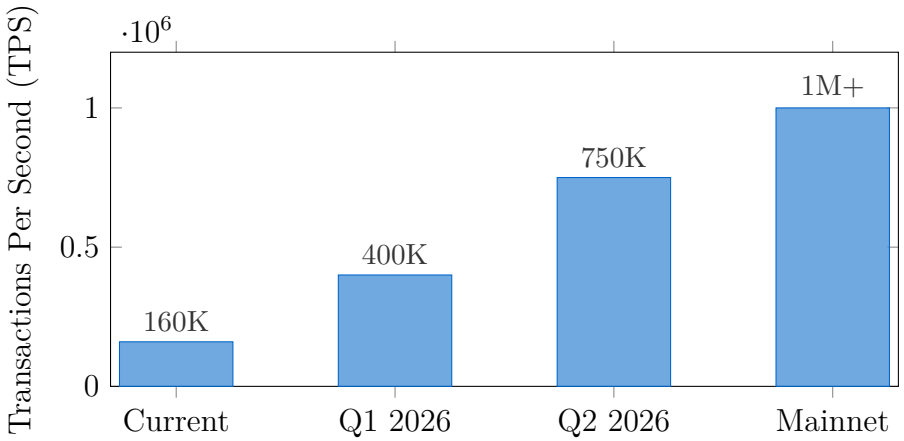
Phase 3: Production Readiness (Q1-Q2 2026)

1. **Tor Enhancements**
   - Advanced circuit management with QRNG seeding
   - Automatic .onion domain registration
   - Traffic analysis resistance improvements
2. **Speed Optimizations**
   - GPU-accelerated consensus verification
   - Parallel block processing
   - Memory-optimized state management
   - Target: 1M+ TPS capacity
3. **Mainnet Preparation**
   - Security audits (external)
   - Economic model finalization
   - Genesis block preparation
   - Launch: December 15, 2026

9.1 Tor Integration Roadmap



9.2 Performance Optimization Targets



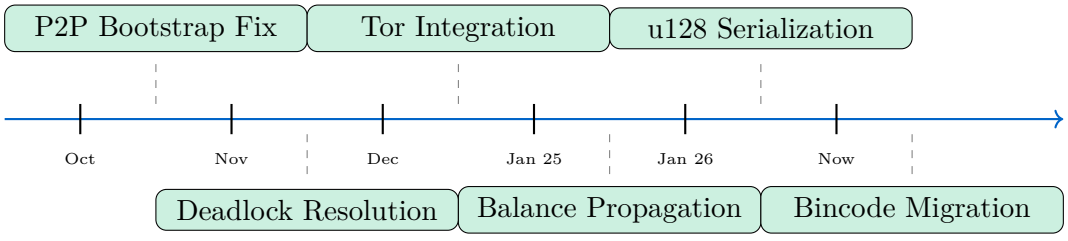
10 Technical Debt & Recent Fixes

10.1 v3.4.15 Critical Fix: CBOR u128 Serialization

× Bug Fixed

Problem: Sync stuck at block 199,000 due to CBOR's inability to serialize u128 values.
Root Cause: BlockPackCodec used CBOR for P2P communication, but CBOR cannot handle u128 integers introduced in the token precision upgrade.
Solution: Migrated BlockPackCodec from CBOR to Bincode serialization, which natively supports u128.
Impact: Full blockchain sync now works correctly past block 199,000.

10.2 Resolved Issues Timeline



11 Conclusion

Q-NarwhalKnight has achieved significant milestones in its development journey, establishing itself as a pioneering quantum-resistant blockchain platform. With over 661,000 lines of production Rust code, 80+ specialized crates, and comprehensive test coverage, the project demonstrates both technical depth and engineering rigor.

11.1 Key Strengths

- **Quantum Readiness**: Production-deployed post-quantum cryptography
- **Privacy by Design**: Full Tor integration with advanced anonymity features
- **Performance**: Targeting 160K+ TPS with path to 1M+ TPS
- **Decentralization**: No central coordinators, pure P2P architecture
- **Code Quality**: Extensive testing, modular architecture

11.2 Path Forward

The immediate focus areas are:

1. **Speed Optimizations**: GPU acceleration, parallel processing
2. **Advanced Tor Features**: QRNG seeding, tor-only mode
3. **Mainnet Preparation**: Security audits, economic finalization

"The future is quantum. The future is decentralized. The future is now."

A Crate Directory

Crate	Lines	Purpose
q-api-server	91,923	REST API, SSE streaming, request handling
q-storage	64,830	RocksDB storage, state management, sync
q-vm	53,447	WASM VM, smart contract execution
q-network	45,392	libp2p networking, P2P protocols
q-tor-client	26,737	Tor integration, circuit management
q-types	17,137	Core type definitions, primitives
q-narwhal-core	9,383	Narwhal mempool, reliable broadcast
q-dag-knight	8,730	DAG-Knight consensus engine
q-crypto-advanced	6,757	Post-quantum cryptography
q-dex	6,004	Decentralized exchange, AMM