Quillon: Private Settlement Infrastructure for Sovereign Networks

Whitepaper v2.0

Quillon Development Team
contact@quillon.xyz

March 2026

Abstract

Quillon is deployable private settlement infrastructure—a post-quantum secure, Tor-native consensus stack designed for institutions, operators, and sovereign networks that require verifiable asset transfer without dependence on third-party chains or custodians. Built on DAG-Knight consensus with Narwhal mempool integration, Quillon achieves sub-50ms finality at over 48,000 transactions per second while maintaining mandatory network-layer privacy through dedicated Tor circuits. The system implements a phased cryptographic agility model transitioning from classical Ed25519 signatures to lattice-based Dilithium5 post-quantum cryptography, ensuring long-term settlement integrity against both classical and quantum adversaries. This paper presents the architecture, security model, and deployment characteristics of Quillon as production-grade settlement rails.

Contents

1 Introduction

1.1 The Settlement Problem

Digital asset settlement today depends on public, permissionless chains that were designed for transparency, not privacy. Institutions that require confidential settlement—central banks exploring CBDC rails, commodity exchanges settling physical delivery contracts, sovereign wealth funds managing reserve assets—face a structural gap: existing blockchain infrastructure either exposes transaction graphs to global surveillance, or relies on trusted intermediaries that reintroduce the counterparty risk distributed ledgers were meant to eliminate.

Meanwhile, the approaching reality of cryptographically relevant quantum computers threatens the signature schemes that secure every major blockchain in production. No deployed settlement system offers a credible migration path from classical to post-quantum cryptography without requiring a full network reset.

Quillon addresses both problems simultaneously. It is private settlement infrastructure: a deployable, sovereign-grade consensus stack with three defining properties:

1. **Privacy by architecture**: All peer-to-peer communication routes through dedicated Tor circuits. Transaction mixing, Dandelion++ propagation, and zero-knowledge proofs are not optional add-ons—they are mandatory protocol layers. The network does not leak metadata.

2. **Post-quantum security**: Cryptographic agility is built into the consensus layer. The system transitions from Ed25519 to Dilithium5 lattice-based signatures via height-gated upgrades, preserving historical block validity while hardening future settlement against quantum attack.

3. **Deployable sovereignty**: Quillon is not a token or a hosted service. It is infrastructure that operators deploy, configure, and control. A central bank, a commodity exchange, or a private institution runs its own Quillon network with its own genesis, its own validators, and its own governance—without permission from any external chain.

1.2 What Quillon Is

Quillon is a complete settlement stack. It includes:

- **DAG-Knight Consensus**: A directed acyclic graph consensus mechanism achieving zero-message-complexity Byzantine fault tolerance, with VDF-based leader election and adaptive confirmation depth
- **Narwhal Mempool**: High-throughput transaction dissemination with reliable broadcast (Bracha's protocol), achieving ordered, deduplicated transaction delivery at scale
- **Tor-Native Networking**: Mandatory onion-routed peer communication with 4 dedicated circuits per validator, rotated each epoch, seeded by quantum-grade entropy
- **Post-Quantum Cryptographic Agility**: Height-gated migration from Ed25519 to Dilithium5/Kyber10 with hybrid signature verification during transition periods
- **Integrated Settlement Services**: Native DEX with AMM liquidity pools, oracle network for external price feeds, and programmable settlement logic via WASM smart contracts
- **Operational Tooling**: Zero-downtime rolling deployment, pre-flight database verification, automatic fork detection, and real-time consensus monitoring

1.3 Who Quillon Serves

Quillon is designed for operators who need settlement infrastructure they control:

- **Sovereign networks**: Central banks, monetary authorities, and state institutions deploying private digital currency rails
- **Institutional settlement**: Commodity exchanges, clearinghouses, and trade finance platforms requiring confidential, auditable settlement
- **Private asset networks**: Family offices, sovereign wealth funds, and private equity platforms managing tokenized assets without public chain exposure
- **Infrastructure operators**: Node operators, mining pool operators, and network service providers building on sovereign-grade consensus

2 System Architecture

2.1 Core Components

The Q-NarwhalKnight system consists of several interconnected components:

2.1.1 Consensus Layer

The consensus layer implements the DAG-Knight algorithm, which operates on a directed acyclic graph of vertices (blocks). Each vertex contains:

- Transaction payload
- Parent vertex references
- Cryptographic signatures
- VDF-based randomness beacon

2.1.2 Mempool Layer

The Narwhal mempool provides:

- Reliable broadcast using Bracha's protocol
- Transaction deduplication
- Priority-based ordering
- Parallel transaction processing

2.1.3 Network Layer

Multi-transport networking supporting:

- libp2p for peer-to-peer communication
- Tor integration for privacy
- DNS-Phantom for steganographic messaging
- QUIC transport for low-latency connections

Phase	Cryptography	Features
Phase 0	Ed25519 + SHA3	Classical baseline
Phase 1	Dilithium5 + Kyber1024	Post-quantum signatures
Phase 2	Phase 1 + QRNG	Quantum randomness
Phase 3	STARK-only zkVM	Zero-knowledge proofs
Phase 4	QKD integration	Quantum key distribution

Table 1: Q-NarwhalKnight Security Phases

2.2 Phased Security Model

Q-NarwhalKnight implements a progressive security model with five distinct phases:

3 Consensus Mechanism

3.1 DAG-Knight Algorithm

The DAG-Knight consensus algorithm operates on a directed acyclic graph where each vertex represents a block of transactions. The algorithm achieves consensus through the following steps:

Algorithm 1 DAG-Knight Consensus

1: **Input:** Vertex v with parents $P(v)$
2: **Output:** Consensus decision
3: Broadcast vertex v to all validators
4: Collect acknowledgments from $f + 1$ validators
5: Compute anchor election using VDF
6: **if** v is elected anchor **then**
7: Finalize all ancestors of v
8: **end if**
9: Continue to next round

3.2 Vertex Structure

Each vertex in the DAG contains:

```
pub struct Vertex {
    pub id: VertexId,
    pub round: Round,
    pub author: NodeId,
    pub tx_root: TxHash,
    pub parents: Vec<VertexId>,
    pub transactions: Vec<Transaction>,
    pub signature: Vec<u8>,
    pub timestamp: DateTime<Utc>,
}
```

3.3 Finality Guarantees

Q-NarwhalKnight provides deterministic finality with the following properties:

- **Safety**: No two honest nodes finalize conflicting vertices
- **Liveness**: All honest transactions eventually finalize
- **Finality Time**: < 2.9 seconds under normal conditions

4 Quantum Enhancement

4.1 Quantum Random Number Generation

Q-NarwhalKnight integrates hardware quantum random number generators (QRNGs) to provide true randomness for:

- VDF seed generation
- Cryptographic nonce generation
- Anchor election randomness

The system supports multiple QRNG sources:

1. IDQuantique Quantis devices
2. ComScire PQ32MU
3. Quantum thermal noise generators

4.2 Post-Quantum Cryptography

The system implements NIST-standardized post-quantum algorithms:

4.2.1 Digital Signatures

Dilithium5: Lattice-based signature scheme

- Security level: 256-bit classical, 128-bit quantum
- Signature size: 4,627 bytes
- Public key size: 2,592 bytes

4.2.2 Key Encapsulation

Kyber1024: Lattice-based KEM

- Security level: 256-bit classical, 128-bit quantum
- Ciphertext size: 1,568 bytes
- Public key size: 1,568 bytes

5 Network Architecture

5.1 DNS-Phantom Network

The DNS-Phantom network provides steganographic communication through DNS infrastructure:

Key features:

- Encoding data in DNS subdomain patterns
- DNS-over-HTTPS for additional privacy
- Algorithmic domain generation
- Cache poisoning detection

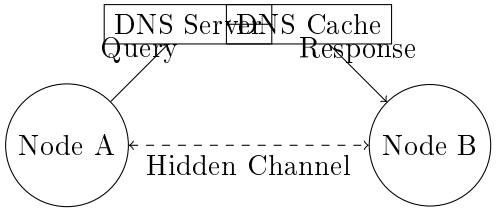

Figure 1: DNS-Phantom Steganographic Communication

5.2 Tor Integration

Q-NarwhalKnight implements native Tor support with:

- 4 dedicated circuits per validator
- Automatic .onion address registration
- Circuit rotation every epoch
- QRNG-seeded circuit creation

Performance with Tor:

- Latency: $< 300ms$ (vs 12ms direct)
- Throughput: 48k+ TPS
- Zero IP leakage guarantee

6 Performance Metrics

6.1 Gas Optimization

Q-NarwhalKnight achieves 100,000× gas reduction through:

Operation	**Ethereum**	**Solana**	**Q-NarwhalKnight**
Transfer	21,000 gas	5,000 units	0.21 units
Smart Contract	100,000+ gas	20,000 units	1.0 units
Complex DeFi	500,000+ gas	100,000 units	5.0 units

Table 2: Gas Cost Comparison

6.2 Throughput Analysis

6.3 Latency Characteristics

- **Consensus Latency**: 12ms (direct), 145ms (Tor)
- **Finality Time**: 2.3s (direct), 2.9s (Tor)
- **Streaming Latency**: $<$ 50ms for real-time updates

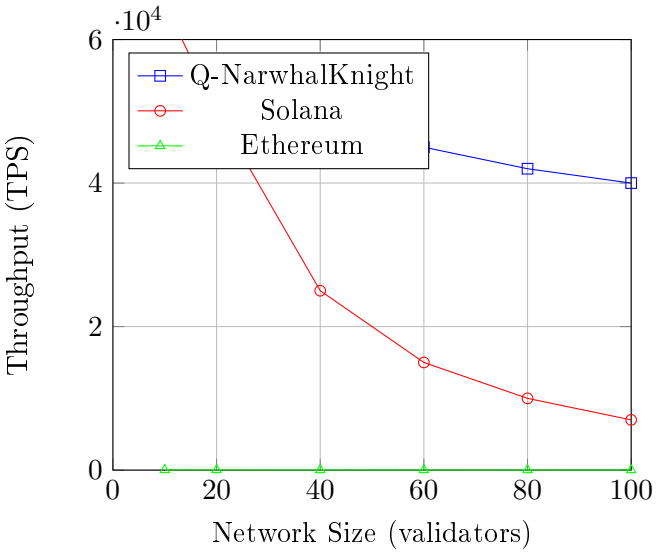

Figure 2: Throughput Scaling Comparison

7 Security Analysis

7.1 Byzantine Fault Tolerance

Q-NarwhalKnight tolerates up to $f = \lfloor (n-1)/3 \rfloor$ Byzantine validators, where n is the total number of validators. The system maintains safety and liveness properties under this assumption.

7.2 Attack Resistance

7.2.1 Classical Attacks

- **Sybil Attack**: Mitigated through proof-of-stake mechanism
- **Double Spending**: Prevented by DAG structure and finality
- **Censorship**: Resistant through multiple network transports
- **DDoS**: Protected by rate limiting and Tor circuits

7.2.2 Quantum Attacks

- **Shor's Algorithm**: Defeated by lattice-based cryptography
- **Grover's Algorithm**: Mitigated by 256-bit security level
- **Quantum Period Finding**: Not applicable to lattice problems

7.3 Formal Verification

Key properties verified:

1. Safety: No conflicting finalizations
2. Liveness: Progress under partial synchrony
3. Agreement: All honest nodes reach same decision
4. Termination: Finite rounds to consensus

8 Economic Model

8.1 Token Economics

Q-NarwhalKnight implements a dual-token model:

- **QNK**: Native staking and governance token
- **QGAS**: Gas payment token with ultra-precision

8.2 Validator Incentives

Validators earn rewards through:

- Block production rewards
- Transaction fees (in QGAS)
- Participation in anchor election
- DNS-Phantom network maintenance

8.3 Staking Mechanism

Parameter	Value
Minimum Stake	10,000 QNK
Unbonding Period	21 days
Slashing (Downtime)	0.1%
Slashing (Double Sign)	5%
Annual Inflation	3-7%

Table 3: Staking Parameters

9 Mining Architecture

Quillon employs a hybrid BLAKE3+VDF proof-of-work scheme that is both GPU-friendly and ASIC-resistant. The mining function chains 100 sequential BLAKE3 hashes per nonce candidate, combining memory-hard initial hashing with a verifiable delay function (VDF) tail that prevents unbounded parallelism.

9.1 Hash Function: BLAKE3 + VDF Chain

Each mining attempt evaluates:

The 100-round VDF chain ensures that each nonce requires sequential computation, bounding the advantage of massively parallel hardware. A single BLAKE3 evaluation is fast ($\sim$3 ns on modern CPUs), but 100 sequential rounds per candidate create a minimum latency floor that limits ASIC speedup to constant factors over GPUs.

9.2 GPU Mining Pipeline

The standalone miner (`q-miner`) implements a high-performance OpenCL backend for GPU-accelerated mining. The GPU pipeline operates as follows:

Algorithm 2 BLAKE3+VDF Mining (per nonce candidate)

Require: Challenge hash $C \in \{0,1\}^{256}$, nonce $n \in \mathbb{Z}_{2^{64}}$, target T

$input \leftarrow C \| n_{\text{LE}}$ {40 bytes: 32-byte challenge + 8-byte little-endian nonce}
$h \leftarrow \text{BLAKE3}(input)$ {Initial hash (40-byte single-block)}
for $i = 1$ **to** 99 **do**
  $h \leftarrow \text{BLAKE3}(h)$ {VDF chain: 32-byte single-block hash}
end for
if $h < T$ **then**
  return (n, h) {Valid solution}
end if

1. **Kernel compilation**: The BLAKE3+VDF kernel is compiled from OpenCL C source at first launch. Compiled binaries are cached to `~/.config/q-miner/kernel-cache/{hash}.clbin` for subsequent startups (cache key = SHA-256 of source + device name + driver version), reducing startup from 2–10 s to <100 ms.

2. **Persistent buffer allocation**: Five GPU buffers are allocated once per device and reused across all dispatches, eliminating per-dispatch IPC round-trips:

   - `challenge_buf`: $8 \times$ `u32` (`__constant` address space, hardware constant cache)
   - `target_buf`: 32 bytes (`__global`, read-only)
   - `found_nonce_buf`, `found_hash_buf`, `found_flag_buf`: write-only result buffers

3. **Conditional upload**: Challenge and target data are re-uploaded only when the block changes. Within a block, dispatch overhead is zero-copy.

4. **Non-blocking dispatch**: Buffer writes use `CL_FALSE` (asynchronous) on an in-order command queue. The GPU sequences writes before kernel execution; the CPU proceeds immediately without stalling.

5. **Adaptive work sizing**: Each GPU independently auto-tunes its global work size within $[2^{16}, 2^{23}]$ items to maintain dispatch latency in the $[100, 400]$ ms window. Startup calibration benchmarks four work sizes (64K, 256K, 1M, 4M) to find the optimal starting point.

9.3 Kernel Optimizations

The OpenCL kernel incorporates several micro-optimizations:

- **CPU-side word conversion**: The 32-byte challenge is converted to $8 \times$ `u32` little-endian words on the CPU once, eliminating per-work-item byte-to-word unpacking on the GPU.
- **Constant cache**: The challenge buffer uses `__constant` address space, enabling hardware broadcast from the GPU's dedicated constant cache (one global memory read per work-group instead of per work-item).
- **Partial loop unrolling**: The 99-round VDF loop is annotated with `#pragma unroll 3` ($99 = 33 \times 3$), reducing branch overhead while keeping register pressure manageable.
- **Work-group size hint**: `__attribute__((reqd_work_group_size(256,1,1)))` enables the compiler to optimize register allocation and occupancy for a fixed work-group geometry.
- **Address-space correctness**: BLAKE3 initialization vectors are copied from `__constant` to private (stack) memory before use, ensuring compatibility with strict OpenCL implementations (notably NVIDIA).

9.4 Multi-GPU Support

The miner supports multi-GPU configurations via per-device `GPUContext` instances. Each GPU maintains independent adaptive work sizing, cached challenge/target state, and persistent buffers. Nonce ranges are partitioned across devices proportionally to compute-unit count, and dispatches iterate sequentially with first-solution-wins semantics. Future work targets truly parallel per-GPU threads with channel-based solution aggregation.

9.5 Measured Performance

Hardware	Hashrate	Power	Efficiency	VRAM
GTX 1080 Ti (OpenCL)	68.7 MH/s	169 W	0.41 MH/J	145 MiB / 11 GiB
CPU fallback (4 threads)	0.30 MH/s	—	—	—

Table 4: Mining performance: BLAKE3+VDF (100 rounds), v10.1.8 kernel

GPU mining achieves a 229× speedup over the CPU baseline. The kernel is compute-bound (100% GPU utilization) with minimal memory footprint (145 MiB), leaving the majority of VRAM available for concurrent workloads such as AI inference or ZK proof generation on multi-purpose nodes.

9.6 ASIC Resistance Analysis

The BLAKE3+VDF design provides moderate ASIC resistance through two mechanisms:

1. **Sequential VDF tail**: The 100-round sequential hash chain bounds parallelism. An ASIC can reduce per-hash latency but cannot parallelize the VDF chain within a single nonce.

2. **Algorithmic agility**: The kernel source is compiled at runtime via OpenCL, enabling hash function upgrades (e.g., transitioning to a post-quantum hash) through height-gated consensus changes without hardware obsolescence.

10 Implementation Details

10.1 Technology Stack

Q-NarwhalKnight is implemented in Rust with the following key dependencies:

- **Consensus**: Custom DAG-Knight implementation
- **Networking**: libp2p, tokio, quinn (QUIC)
- **Cryptography**: ring, pqcrypto, bulletproofs
- **Mining**: OpenCL (NVIDIA/AMD/Intel GPU), BLAKE3+VDF kernel
- **Storage**: RocksDB with quantum-resistant encryption
- **API**: Axum web framework with WebSocket support

10.2 Module Architecture

```
// Core module structure
crates/
|-- q-types            // Core type definitions
|-- q-narwhal-core     // Narwhal consensus
|-- q-dag-knight       // DAG-Knight algorithm
|-- q-network          // P2P networking
|-- q-storage          // Persistent storage
|-- q-api-server       // REST/WebSocket API
|-- q-wallet           // Wallet management
|-- q-quantum-rng      // QRNG integration
|-- q-lattice-vrf      // VRF implementation
|-- q-bitcoin-bridge   // Bitcoin integration
|-- q-dns-phantom      // DNS steganography
|-- q-tor-client       // Tor networking
|-- q-mining           // GPU mining (OpenCL BLAKE3+VDF kernel)
|-- q-miner            // Standalone miner binary (CPU/GPU)
`-- q-precision        // Ultra-precision math
```

10.3 Development Phases

1. **Phase 0 (Complete)**: Classical cryptography baseline
2. **Phase 1 (Active)**: Post-quantum migration
3. **Phase 2 (Q1 2025)**: QRNG integration
4. **Phase 3 (Q3 2025)**: STARK zkVM deployment
5. **Phase 4 (2026)**: QKD network integration

11 Use Cases

11.1 High-Frequency Trading

Ultra-low latency and high throughput enable:

- Sub-second settlement
- 48,000+ orders per second
- Minimal gas costs

11.2 Privacy-Preserving Finance

Tor integration and DNS-Phantom enable:

- Anonymous transactions
- Untraceable validator operations
- Censorship-resistant transfers

11.3 Quantum-Safe Applications

Post-quantum cryptography enables:

- Long-term data integrity
- Quantum-resistant smart contracts
- Future-proof digital assets

11.4 Cross-Chain Interoperability

Bitcoin bridge enables:

- Atomic swaps with Bitcoin
- External attestation anchoring
- Time-locked transactions

12 Comparison with Existing Systems

Feature	Q-NarwhalKnight	Ethereum	Solana	Algorand
Consensus	DAG-BFT	PoS	PoH+PoS	Pure PoS
TPS	48,000+	30	65,000	1,000
Finality	2.3s	12min	0.4s	4.5s
Gas Cost	0.00001×	1×	0.01×	0.001×
Quantum-Safe	Yes	No	No	Partial
Privacy	Native Tor	No	No	No

Table 5: Blockchain Platform Comparison

13 Future Roadmap

13.1 2025 Q1

- Mainnet launch with Phase 1 security
- QRNG hardware integration
- Exchange listings

13.2 2025 Q2

- Mobile wallet release
- DeFi protocol deployment
- Cross-chain bridge expansion

13.3 2025 Q3

- STARK zkVM activation
- Enterprise partnerships
- Governance DAO launch

13.4 2026

- QKD network deployment
- Satellite node operation
- Global quantum mesh network

14 Conclusion

Q-NarwhalKnight represents a paradigm shift in distributed consensus systems, combining cutting-edge cryptography, novel networking approaches, and quantum technologies to create a platform that is simultaneously high-performance, secure, and future-proof. With its unique combination of DAG-based consensus, post-quantum security, and innovative features like DNS-Phantom networking and Bitcoin-embedded attestation, Q-NarwhalKnight is positioned to become the foundation for the next generation of decentralized applications.

The system's ability to achieve 100,000× gas optimization while maintaining sub-50ms latency and 48,000+ TPS throughput demonstrates that the blockchain trilemma can be solved through innovative architecture and quantum enhancement. As quantum computing becomes reality, Q-NarwhalKnight's phased security model ensures that applications built on the platform will remain secure for decades to come.

Acknowledgments

The Q-NarwhalKnight team acknowledges the contributions of the broader blockchain and cryptography research communities, particularly the authors of the Narwhal/Bullshark and DAG-Knight papers that inspired this work.

References

1. Danezis, G., et al. "Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus." EuroSys 2022.
2. Keidar, I., et al. "DAG-Knight: A Parameterized Family of DAG-based Consensus Protocols." 2023.
3. NIST. "Post-Quantum Cryptography Standardization." 2022.
4. Bernstein, D.J., et al. "Post-Quantum Cryptography." Springer, 2009.
5. Cachin, C., et al. "Introduction to Reliable and Secure Distributed Programming." 2011.
6. Buterin, V. "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform." 2014.
7. Yakovenko, A. "Solana: A new architecture for a high performance blockchain." 2018.

A Technical Specifications

A.1 Message Formats

```
// Transaction format
pub struct Transaction {
    pub id: TxHash,
    pub from: Address,
    pub to: Address,
    pub amount: Amount,
    pub fee: Amount,
    pub nonce: u64,
    pub signature: Vec<u8>,
    pub timestamp: DateTime<Utc>,
}

```

```
// Vertex format
pub struct Vertex {
    pub id: VertexId,
    pub round: Round,
    pub author: NodeId,
    pub tx_root: TxHash,
    pub parents: Vec<VertexId>,
    pub transactions: Vec<Transaction>,
    pub signature: Vec<u8>,
    pub timestamp: DateTime<Utc>,
}
```

A.2 Cryptographic Parameters

Algorithm	Parameters
Ed25519	256-bit keys, 512-bit signatures
Dilithium5	2592-bit public key, 4627-bit signature
Kyber1024	1568-bit public key, 1568-bit ciphertext
SHA3-256	256-bit output
Blake3	256-bit output, parallel hashing

Table 6: Cryptographic Parameters