Quasar: Quantum-Secure Multi-Engine Consensus with Triple-Proof Quantum Finality

Zach Kelling

Lux Industries

2022 (Revised 2025)

Abstract

We present **Quasar**, a quantum-secure consensus protocol family for Lux Network's Q-Chain, achieving **10 ms finality** with GPU-accelerated BLS (100 ms CPU BLS fallback) through triple-proof validation combining classical BLS signatures with post-quantum Corona threshold signatures. Quasar consists of six layered consensus engines (Photon, Wave, Nova, Nebula, Prism, Quasar) supporting both linear chains and DAGs, integrated with Verkle trees for efficient state proofs and witness validation. The triple-proof mechanism creates a <10 ms attack window physically impossible to exploit even with large-scale quantum computers, while maintaining performance competitive with classical consensus systems. Q-Chain replaces Lux 1.0's P-Chain as the platform management layer, handling validator coordination, staking operations, appchain creation, and network governance with quantum-resistant guarantees. With a 1 B gas block limit (∼47,619 txs/block), Quasar achieves **4,761,904 TPS with finality** using GPU BLS, providing defense-in-depth against both classical and quantum adversaries.

1 Introduction

Blockchain consensus protocols face an existential challenge from quantum computing. Shor's algorithm can break elliptic curve signatures in polynomial time [1], threatening the security of all ECDSA and BLS-based blockchains. While post-quantum cryptography standards have emerged [2], integrating them without sacrificing performance remains unsolved.

1.1 The Quantum Threat Timeline

- **2030-2035**: NIST estimates quantum computers capable of breaking RSA-2048 and ECDSA [3]
- **Harvest-now-decrypt-later**: Adversaries store encrypted blockchain data today, decrypt later with quantum computers
- **<10ms attack window**: Even theoretical quantum computers cannot break BLS12-381 in the narrow GPU BLS finality window we achieve

1.2 Quasar's Solution

Quasar addresses quantum threats through:

1. **Triple-proof finality**: Require both classical (BLS) and post-quantum (Corona) signatures for block finalization
2. **Narrow attack window**: 10 ms GPU BLS finality leaves no time for quantum attacks
3. **Modular architecture**: Six consensus engines supporting different blockchain types (linear, DAG, voting)
4. **Efficient proofs**: Verkle trees and witness validation for scalable state verification

2 System Architecture

2.1 Quasar Consensus Stack

The Quasar family consists of six layered protocols:

Engine	Purpose	Complexity
Photon	Binary consensus	O(K × Beta)
Wave	Threshold consensus	O(K × choices × Beta)
Nova	DAG finalization	O(vertices × K)
Nebula	Full DAG consensus	O(vertices^2 × K)
Prism	Direct voting	O(N)
Quasar	Quantum overlay	O(3N) for triple-proof

2.2 Triple-Proof Architecture

```
Block Proposal (1ms target)
     |
     |--- BLS Path (per-block) ---------$\blacktriangleright$ 48 bytes aggregated
     |     |--$\blacktriangleright$ GPU aggregation (~3ms)
     |     |--$\blacktriangleright$ Per-block: hash ref (32 bytes)
     |
     |--- PQ Path (per-epoch, 1s) ------$\blacktriangleright$ 192 bytes Groth16 proof
     |     |--$\blacktriangleright$ ML-DSA signing (~495us/validator)
     |     |--$\blacktriangleright$ Corona threshold (~7ms)
     |     |--$\blacktriangleright$ Groth16 prover (~400ms CPU / ~5-15ms GPU)
     |
     |--- Epoch Cert (every 1000 blocks)
           BLS aggregate:  48 bytes
           ZK proof:      192 bytes (Groth16, circuit-independent)
           Total:         ~248 bytes quantum finality
```

3 Core Innovation: Triple-Proof Quantum Finality

3.1 The Triple-Proof Mechanism

Q-Chain requires three cryptographic proofs for quantum finality:

1. BLS Aggregated Signature (Classical)

- BLS12-381 curve with 128-bit classical security
- Aggregatable signatures for efficiency
- 48-byte public keys, 96-byte signatures
- Compatible with existing infrastructure

2. Corona Threshold Signature (Post-Quantum, Privacy)

- Lattice-based (Module-LWE, Ringtail-derived) with 128-bit post-quantum security
- Threshold scheme: no single validator holds full key
- Two-round protocol for efficiency

- Provides anonymous validator participation (ring signature anonymity set)
- Embedded in the ZK proof—zero additional block header bytes

3. ZK Proof of ML-DSA Identities (Post-Quantum, Identity)

- STARK proof aggregating N ML-DSA-65 (FIPS 204) signatures into ∼200 bytes
- Each validator signs with ML-DSA (3,309 byte signature) per epoch
- STARK compression: $N \times 3{,}309$ bytes $\rightarrow$ ∼200 bytes
- Provides post-quantum identity binding—even if BLS is broken, validator identities remain provable
- QuasarCert verification: ∼2–5 ms CPU (dominated by Groth16's 3 pairings on BLS12-381) [10]

1: **function** IsBlockFinal(block, cert)
2: $valid_{BLS} \leftarrow$ VerifyBLS($cert.BLS, block$)
3: $valid_{ZK} \leftarrow$ VerifySTARK($cert.ZKProof, block, cert.Validators$)
4: **return** $valid_{BLS} \wedge valid_{ZK}$
5: **end function**

3.1.1 Storage Efficiency at 1ms Block Times

Component	Size	Frequency	Annual
Per-block BLS hash	32 B	1,000/s	∼1.0 TB
Per-epoch triple proof	248 B	1/s	∼7.8 GB
Total (headers + proofs)			**∼1.0 TB**

3.2 Security Analysis

The triple-proof design provides defense in depth:

Attack Scenario	BLS Cert	Corona Cert	Result
Classical Attacker	Secure (128-bit)	Secure (harder)	✓Block Safe
Quantum Attacker	Vulnerable	Secure (128-bit PQ)	✓Block Safe
BLS Implementation Bug	Compromised	Secure	✓Block Safe
Corona Bug	Secure	Compromised	✓Block Safe
Both Compromised	Compromised	Compromised	× Block Unsafe

3.3 Quantum Attack Window

Q-Chain's rapid finality creates an impossibly narrow attack window:

$$\text{Attack Window} < 10\text{ms (GPU BLS)},\ < 100\text{ms (CPU BLS fallback)} \tag{1}$$

$$\text{Quantum Operations Required} > 10^{12} \text{ (for BLS12-381)} \tag{2}$$

$$\text{Available Time} \ll \text{Required Time} \tag{3}$$

Even with a 10,000-qubit quantum computer running optimal Shor's algorithm, breaking BLS12-381 would require billions of sequential operations, far more than possible in 10 ms.

4 Consensus Engines

4.1 Photon: Sampling-Based Consensus

Binary consensus using network sampling:

function QUERYROUND(preference, validators)
$sample \leftarrow$ RandomSample($validators, K$)
$votes \leftarrow$ QueryPreference($sample$)
if $votes \geq \alpha$ **then**
    $confidence \leftarrow confidence + 1$
    if $confidence \geq \beta$ **then**
        return `FINALIZED`
    end if
else
    $confidence \leftarrow 0$
end if
return `CONTINUE`
end function

Parameters:

- $K = 25$: Sample size
- $\alpha = 15$: Quorum threshold
- $\beta = 20$: Confidence threshold

4.2 Wave: Thresholding Consensus

Fast finality through adaptive thresholding:

function WAVEQUERY(choices, validators)
$sample \leftarrow$ RandomSample($validators, K$)
$votes \leftarrow$ QueryPreferences($sample, choices$)
for each $choice \in choices$ **do**
    if $votes[choice] \geq \alpha$ **then**
        $preferences[choice] \leftarrow preferences[choice] + 1$
        if $preferences[choice] \geq \beta$ **then**
            return $choice$ ▷ Finalized
        end if
    end if
end for
return `CONTINUE`
end function

4.3 Nova: DAG Finalizer

Finalizes transactions in DAG structures using Verkle proofs:

function FINALIZEVERTEX(vertex, dag)
$proof \leftarrow$ GenerateVerkleWitness($vertex$)
if ValidateWithWitness($vertex, proof$) **then**

```
    dag.Finalize(vertex)
    return TRUE
end if
return FALSE
end function
```

Verkle Tree Benefits:

- $O(\log n)$ proof size vs. $O(n)$ for Merkle trees
- Constant-time verification
- Efficient state witness generation

4.4 Nebula: Full DAG Consensus

Complete DAG consensus with parallel transaction processing:

```
function ProcessTransaction(tx, dag)
witness ← witnessCache.Get(tx.ID)
if verkleTree.ValidateWithWitness(tx, witness) then
    dag.AddVertex(tx)
    Broadcast(tx)                                   ▷ Parallel propagation
    return TRUE
end if
return FALSE
end function
```

4.5 Prism: Voting-Based Consensus

Direct voting for governance operations:

```
function ProcessVote(vote, proposal)
votes[proposal][vote.NodeID] ← vote
support ← CalculateSupport(proposal)
if support ≥ threshold then
    ExecuteProposal(proposal)
    return APPROVED
end if
return PENDING
end function
```

4.6 Quasar: Quantum-Secure Overlay

The pinnacle layer adding triple-proof finality:

```
function FinalizeBlock(block)
Launch CollectBLS(block) in parallel
Launch CollectCorona(block) in parallel
Wait for both with timeout = 10ms (GPU BLS) / 100ms (CPU fallback)
if both certificates valid then
    return TripleProof(blsCert, rtCert)
else
```

```
    return TIMEOUT                                          ▷ Retry collection
end if
end function
```

5 Platform Management

As the successor to P-Chain in Lux 2.0, Q-Chain handles all platform management with quantum-secure guarantees:

5.1 Validator Management

- **Minimum Stake**: 2,000 LUX
- **Delegation**: Support for delegated staking with customizable fees
- **Rewards**: Automatic distribution with quantum-secure signatures
- **Slashing**: Quantum-resistant penalty mechanisms

5.2 Appchain Creation and Management

```
function CreateAppchain(owners, threshold, controlKeys)
blsSig ← SignBLS(owners, controlKeys)
rtSig ← SignCorona(owners, controlKeys)
tripleProof ← TripleProof(blsSig, rtSig)
if Verify(tripleProof) then
    appchain ← AllocateAppchain(owners, threshold)
    return appchain
end if
return INVALID
end function
```

6 Performance Characteristics

6.1 Mainnet Configuration (21 validators)

Parameter	Value
K (Sample size)	21
α (Preference quorum)	13
α_{conf} (Confidence quorum)	18
β (Confidence threshold)	8
Q-Threshold (Corona)	15 of 21
Quasar Timeout	10ms (GPU BLS) / 100ms (CPU fallback)
Block Gas Limit	1,000,000,000 (1 B)
Txs/Block	∼47,619 (at 21,000 gas/tx)
Finality Target	10ms (GPU BLS) / 100ms (CPU BLS fallback)

6.2 Performance Metrics

Metric	Value	Description
Block Gas Limit	1 B	47,619 txs/block at 21k gas/tx
Finality (GPU BLS)	10ms	Primary target (co-located validators)
Finality (CPU BLS)	100ms	Fallback (commodity hardware)
GPU BLS Aggregation	3ms	Metal/CUDA shader
Corona Aggregation	7ms	PQ signature combination
Certificate Size	2.9KB	Combined BLS + Corona

6.3 Throughput Analysis

Under Byzantine conditions ($f < n/3$), with 1 B gas block limit:

$$\text{Txs/block} = \frac{1{,}000{,}000{,}000}{21{,}000} \approx 47{,}619 \tag{4}$$

$$\text{TPS (GPU BLS)} = \frac{47{,}619}{0.010\text{s}} = 4{,}761{,}904 \text{ TPS} \tag{5}$$

$$\text{TPS (CPU BLS)} = \frac{47{,}619}{0.100\text{s}} = 476{,}190 \text{ TPS} \tag{6}$$

Finality probability after β rounds:

$$P(\text{finality}) \geq 1 - \epsilon, \quad \epsilon \approx 10^{-10} \tag{7}$$

7 Post-Quantum Security

7.1 Corona Threshold Signatures

Corona provides quantum resistance based on lattice problems [4]:

Parameter	Value
Lattice Dimension	1024
Security Level	128-bit post-quantum
Ring Modulus	$2^{32} - 5$
Error Distribution	Gaussian $\sigma = 3.2$
Share Size	1KB
Combination Time	7ms (15-of-21)

7.2 Two-Round Protocol

Round 1: Share Generation

$$\text{share}_i = \text{Lattice-Sign}(sk_i, message) \tag{8}$$

$$\text{time} \approx 48\text{ms (network-bound)} \tag{9}$$

Round 2: Share Combination

$$\sigma = \text{Combine}(\{\text{share}_i\}_{i \in S}), \quad |S| \geq t \tag{10}$$

$$\text{time} \approx 7\text{ms (computation)} \tag{11}$$

8 Security Considerations

8.1 Byzantine Fault Tolerance

Q-Chain maintains safety under standard Byzantine assumptions:

Theorem 1 (Safety)**.** *If $f < n/3$ validators are Byzantine and the network delay $\Delta < \Delta_{max}$, then no two honest validators finalize conflicting blocks.*

Proof. For a block to finalize, it requires:

1. BLS signatures from $\geq 2n/3$ validators
2. Corona shares from $\geq 2n/3$ validators
3. Confidence $\geq \beta$ in Lux voting

With $f < n/3$ Byzantine nodes, at least $n - f > 2n/3$ honest nodes exist. Two conflicting blocks cannot both obtain $2n/3$ signatures from honest validators. □

8.2 Liveness

Theorem 2 (Liveness)**.** *If $f < n/3$ validators are Byzantine and network delay $\Delta < \Delta_{max}$, then all valid transactions eventually finalize.*

Proof. Honest validators always prefer valid transactions. With $> 2n/3$ honest validators and bounded network delay, the Lux consensus mechanism guarantees that valid preferences reach confidence threshold β within finite rounds. □

8.3 Slashing Conditions

Reason	**Evidence**	**Penalty**
Double Sign	Two conflicting block sigs	100% stake
Missing PQ Cert	No Corona signature	50% stake
Invalid Signature	Malformed signature	75% stake
Extended Downtime	¿99% missed blocks	25% stake

9 Network Deployment

9.1 Multi-Chain Architecture

Q-Chain can secure multiple blockchains simultaneously with different consensus configurations:

Chain Type	Engine	K	Finality
Financial (High Security)	Quasar+Wave	30	15ms (GPU) / 120ms (CPU)
Gaming (High Throughput)	Quasar+Photon	15	10ms (GPU) / 80ms (CPU)
Governance (Voting)	Quasar+Prism	21	10ms (GPU) / 100ms (CPU)
DeFi (Balanced)	Quasar+Nebula	25	12ms (GPU) / 110ms (CPU)

10 Implementation

10.1 Directory Structure

```
/quasar/
|---- consensus/        # Core algorithms
|   |---- photon/       # Binary consensus
|   |---- wave/         # Multi-choice consensus
|   |---- nova/         # DAG finalizer
|   |---- nebula/       # Full DAG consensus
|   |---- prism/        # Direct voting
|   |---- quasar/       # Quantum overlay
|---- crypto/           # Cryptographic primitives
|   |---- bls/          # BLS12-381 operations
|   |---- corona/     # Post-quantum threshold
|---- verkle/           # Verkle tree implementation
|---- validators/       # Validator management
|---- slashing/         # Economic penalties
```

10.2 Performance Optimization

Parallel Certificate Collection:

- BLS and Corona collection run concurrently
- Non-blocking network I/O with timeout
- Early termination on quorum

Verkle Tree Caching:

- LRU cache for witness proofs
- Batch witness generation
- Incremental tree updates

11 Future Work

11.1 Dynamic Validator Sets

- Hot-swapping validators without downtime
- Rapid DKG for new Corona keys
- Forward-secure key evolution

11.2 Cross-Chain Atomic Operations

- Leverage triple-proof finality for atomic swaps
- Quantum-safe hash time-locked contracts
- Inter-chain certificate validation

11.3 Light Client Support

- Succinct triple-proof proofs
- Post-quantum Merkle trees
- Mobile-friendly verification

11.4 Hardware Integration

- HSM support for key protection
- Hardware-accelerated lattice operations
- TEE integration for share generation

12 Appendix A: Consensus Parameter Tuning

12.1 Safety vs. Liveness Trade-offs

Increasing α and β improves safety at the cost of latency:

α	β	**Safety**	**Finality Latency**
13	8	99.9999%	8ms (GPU) / 80ms (CPU)
15	10	99.99999%	10ms (GPU) / 100ms (CPU)
18	12	99.999999%	12ms (GPU) / 120ms (CPU)

12.2 Network Size Scaling

Optimal K grows with network size:

$$K_{opt} \approx \sqrt{N} \tag{12}$$

$$\alpha \approx 0.6 \times K \tag{13}$$

$$\beta \approx 0.4 \times K \tag{14}$$

13 Appendix B: Cryptographic Specifications

13.1 BLS12-381 Parameters

- Curve: $y^2 = x^3 + 4$ over $\mathbb{F}_p$

- Embedding degree: 12
- Subgroup size: 381 bits
- Security level: 128-bit classical

13.2 Corona Parameters

- Lattice: LWE with dimension 1024
- Modulus: $q = 2^{32} - 5$
- Error: Discrete Gaussian with $\sigma = 3.2$
- Security: 128-bit post-quantum (NIST Level III)

14 Conclusion

Quasar represents a fundamental advancement in blockchain consensus design, achieving quantum security without sacrificing performance. Through triple-proof finality combining classical BLS with post-quantum Corona signatures, Q-Chain provides:

1. **10 ms GPU BLS finality** (100 ms CPU fallback) with triple-proof quantum security
2. **4,761,904 TPS** with finality at 1 B gas block limit
3. **Quantum resistance** through defense-in-depth
4. **Modular architecture** supporting various blockchain types
5. **Physical impossibility** of real-time quantum attacks (<10 ms window)

The narrow <10 ms attack window with GPU BLS makes quantum attacks physically impossible, while the modular consensus stack (Photon, Wave, Nova, Nebula, Prism, Quasar) provides flexibility for different use cases. Q-Chain positions Lux Network at the forefront of blockchain security for the next generation of decentralized applications.

By combining sampling-based consensus, threshold cryptography, and post-quantum signatures, Quasar achieves the seemingly impossible: quantum security with classical-level performance.

References

[1] Shor, P.W. (1994). *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. SIAM Journal on Computing, 26(5), 1484-1509.

[2] NIST (2024). *Post-Quantum Cryptography Standardization*. National Institute of Standards and Technology.

[3] Mosca, M. (2018). *Cybersecurity in an Era with Quantum Computers*. IEEE Security & Privacy, 16(5), 38-41.

[4] Lux Industries (2026). *Corona: Two-Round Module-LWE Threshold Signatures*. `github.com/luxfi/corona`. Lux's own Module-LWE (Ringtail-derived) two-round threshold scheme; builds on the Ringtail line [5].

[5] C. Boschini, D. Kaviani, R. W. F. Lai, G. Malavolta, A. Takahashi, and M. Tibouchi (2024). *Ringtail: Practical Two-Round Threshold Signatures from LWE*. Cryptology ePrint Archive, Paper 2024/1113; IEEE S&P 2025. Academic two-round lattice threshold (IACR ePrint 2024/1113).

[6] Boneh, D., Lynn, B., & Shacham, H. (2001). *Short Signatures from the Weil Pairing*. Advances in Cryptology—ASIACRYPT 2001, 514-532.

[7] Kuszmaul, J. (2019). *Verkle Trees*. Ethereum Research.

[8] Team Rocket (2020). *Scalable and Probabilistic Leaderless BFT Consensus through Metastability*. arXiv:1906.08936.

[9] Blackshear, S. et al. (2022). *Narwhal and Tusk: A DAG-based Mempool and Efficient BFT Consensus*. EuroSys 2022.

[10] Lux Industries (2026). *Quasar Consensus — Measured Benchmarks*. Apple M1 Max, darwin/arm64, Go 1.26.1. `https://github.com/luxfi/consensus/blob/main/BENCHMARKS.md`.