Cross-Chain Bridge Protocol

Trustless Atomic Swaps with Multi-Sig Committee Validation for Wrapped Bitcoin, Zcash, Iron Fish, and Ethereum

Q-NarwhalKnight Whitepaper Series

Version 2.0 — February 2026

Authors: Q-NarwhalKnight Foundation Research Team

https://quillon.xyz

Abstract

We present Version 2.0 of the Q-NarwhalKnight Cross-Chain Bridge Protocol, a trustless, non-custodial system enabling atomic swaps between Q-NarwhalKnight's quantum-resistant blockchain and four major chains: **Bitcoin** (BTC), **Zcash** (ZEC), **Iron Fish** (IRON), and **Ethereum** (ETH). The protocol introduces wrapped token representations—wBTC, wZEC, wIRON, and wETH—that are minted on deposit and burned on withdrawal, maintaining strict 1:1 collateral backing at all times. Version 2.0 introduces a **decentralized multi-signature bridge committee**: a rotating panel of 11 validator nodes that independently verify every bridge claim, requiring a 7-of-11 threshold before any mint or burn operation proceeds. Committee membership is deterministically selected via SHA3(block_hash‖epoch‖peer_id) and rotates every 100 blocks, preventing collusion and single-point-of-failure risks. The protocol retains Hash Time-Locked Contracts (HTLCs) with staggered timelocks for atomicity, steganographic node discovery on Bitcoin, and Tor-routed communications for censorship resistance. All wrapped tokens are first-class citizens on Q-NarwhalKnight's DEX with sub-3-second finality through DAG-Knight consensus.

Contents

1 Introduction

The proliferation of heterogeneous blockchains has created a fragmented landscape where value remains siloed within individual networks. Bitcoin's $1.8 trillion market capitalization, Ethereum's $400 billion smart contract ecosystem, Zcash's pioneering zero-knowledge privacy, and Iron Fish's encrypted-by-default architecture each represent distinct value propositions that users should be able to access without sacrificing sovereignty or security.

Q-NarwhalKnight's DAG-Knight consensus provides sub-3-second finality and quantum-resistant cryptography through its hybrid Ed25519/Dilithium5 signature scheme. By extending this infrastructure with a trustless cross-chain bridge, we unlock five critical capabilities:

1. **BTC and ETH Liquidity on QNK:** Bitcoin and Ethereum holders can bring liquidity into Q-NarwhalKnight's AMM pools without trusted intermediaries.
2. **Privacy-Preserving Cross-Chain Swaps:** Zcash's shielded transactions and Iron Fish's encrypted-by-default model are preserved during the bridge process.
3. **Quantum-Resistant Custody:** Once bridged, assets benefit from Q-NarwhalKnight's post-quantum cryptographic protections.
4. **Unified DEX Trading:** All wrapped assets trade on a single AMM with deep liquidity across wBTC/QUG, wZEC/QUG, wIRON/QUG, and wETH/QUG pairs.
5. **Decentralized Validation:** A rotating 7-of-11 multi-signature committee independently verifies every bridge claim, eliminating single-point-of-failure risks inherent in centralized bridge designs.

1.1 Design Principles

The bridge protocol adheres to five non-negotiable design principles:

Non-Custodial
No third party ever holds user private keys. All swaps are cryptographically enforced.

Atomic
Swaps either complete fully on both chains or fully refund. Partial execution is mathematically impossible.

1:1 Backed
Every wrapped token in circulation is collateralized by exactly one unit of the native asset locked in an HTLC.

Privacy-First
Tor routing, steganographic discovery, and shielded transaction support prevent metadata leakage.

Censorship-Resistant
No single point of failure; the Bitcoin blockchain itself serves as an immutable bulletin board for node discovery.

2 Wrapped Token Architecture

2.1 Token Specifications

Each bridge chain maps to a dedicated wrapped token on Q-NarwhalKnight with a deterministic 32-byte address derived from the ASCII encoding of its symbol.

Table 1: Wrapped Token Specifications

Symbol	Name	Native	Decimals	Base Unit	Address Derivation
wBTC	Wrapped Bitcoin	BTC	8	Satoshi	`0x77425443...00`
wZEC	Wrapped Zcash	ZEC	8	Zatoshi	`0x775A4543...00`
wIRON	Wrapped Iron Fish	IRON	8	Ore	`0x7749524F4E...00`
wETH	Wrapped Ethereum	ETH	18	Wei	`0x77455448...00`

All three tokens use 8-decimal precision, matching their native chain denominations. This eliminates rounding errors during the bridge process and ensures that 1 BTC $= 10^8$ satoshis maps exactly to 1 wBTC $= 10^8$ base units on Q-NarwhalKnight.

2.2 Address Derivation

Token addresses are 32-byte arrays constructed by encoding the symbol in ASCII and zero-padding:

```
wBTC:  [0x77, 0x42, 0x54, 0x43, 0x00, ..., 0x00]  // "wBTC" + 28 zero bytes
wZEC:  [0x77, 0x5A, 0x45, 0x43, 0x00, ..., 0x00]  // "wZEC" + 28 zero bytes
wIRON: [0x77, 0x49, 0x52, 0x4F, 0x4E, 0x00, ..., 0x00]  // "wIRON" + 27 zero bytes
```

This deterministic scheme ensures that any node can independently compute the canonical address for a bridge token without coordination, and that addresses are human-readable when hex-decoded.

2.3 Balance Storage

Wrapped token balances are stored in Q-NarwhalKnight's unified token balance system:

$$\texttt{token\_balances} : (\underbrace{\texttt{wallet}}_{\text{32 bytes}}, \underbrace{\texttt{token\_addr}}_{\text{32 bytes}}) \longrightarrow \underbrace{\texttt{balance}}_{\text{u128}} \tag{1}$$

Balances are maintained in a concurrent in-memory `HashMap` for low-latency reads and batch-synced to RocksDB every 15 seconds. Bridge operations (mint/burn) trigger *immediate* persistence to prevent data loss on crash—a lesson learned from prior token balance persistence bugs (see Section 11).

3 Hash Time-Locked Contract Protocol

3.1 Protocol Overview

The bridge uses a classic HTLC atomic swap protocol extended with staggered timelocks and privacy enhancements. The protocol involves two parties:

- **User**: Wants to exchange native coins (BTC/ZEC/IRON) for QNK assets or vice versa.
- **Bridge Mediator**: An automated protocol agent that locks matching funds and facilitates the atomic swap.

3.2 HTLC Script Construction

On the Bitcoin side, the HTLC is implemented as a Pay-to-Witness-Script-Hash (P2WSH) script:

Bitcoin HTLC Script (P2WSH)

```
OP_IF
    OP_SHA256 <hash_lock> OP_EQUALVERIFY
    <recipient_pubkey> OP_CHECKSIG
OP_ELSE
    <timelock_blocks> OP_CHECKLOCKTIMEVERIFY OP_DROP
    <refund_pubkey> OP_CHECKSIG
OP_ENDIF
```

This script provides exactly two spending conditions:

Claim Path: The recipient provides the SHA-256 preimage s such that $H(s) = \texttt{hash\_lock}$, plus a valid signature.

Refund Path: After `timelock_blocks` have passed, the original sender can reclaim funds with their signature.

3.3 Cryptographic Primitives

Secret Generation: A cryptographically secure 256-bit random secret $s \xleftarrow{R} \{0,1\}^{256}$ is generated.

Hash Lock: The hash lock $h = \text{SHA-256}(s)$ binds both sides of the swap.

Security Level: With 256-bit preimage resistance, brute-forcing s from h requires $\mathcal{O}(2^{256})$ operations, exceeding even post-quantum security margins.

3.4 Staggered Timelock Architecture

A critical safety requirement is that timelocks are *staggered* across chains to prevent race conditions:

Table 2: Timelock Parameters by Chain

Chain	Timelock	Approx. Duration	Mechanism
Bitcoin	144 blocks	∼24 hours	`OP_CHECKLOCKTIMEVERIFY`
Q-Network	12 hours	12 hours	UTC timestamp comparison
Zcash	120 blocks	∼2.5 hours	Block height
Ethereum	720 blocks	∼2.4 hours	`block.timestamp`
Iron Fish	60 blocks	∼60 minutes	Block height

The staggering ensures that the Q-Network timeout always fires *before* the native chain timeout, giving the bridge mediator time to claim native-chain funds using the revealed secret before the user can attempt a double-claim via refund.

3.5 Protocol Sequence

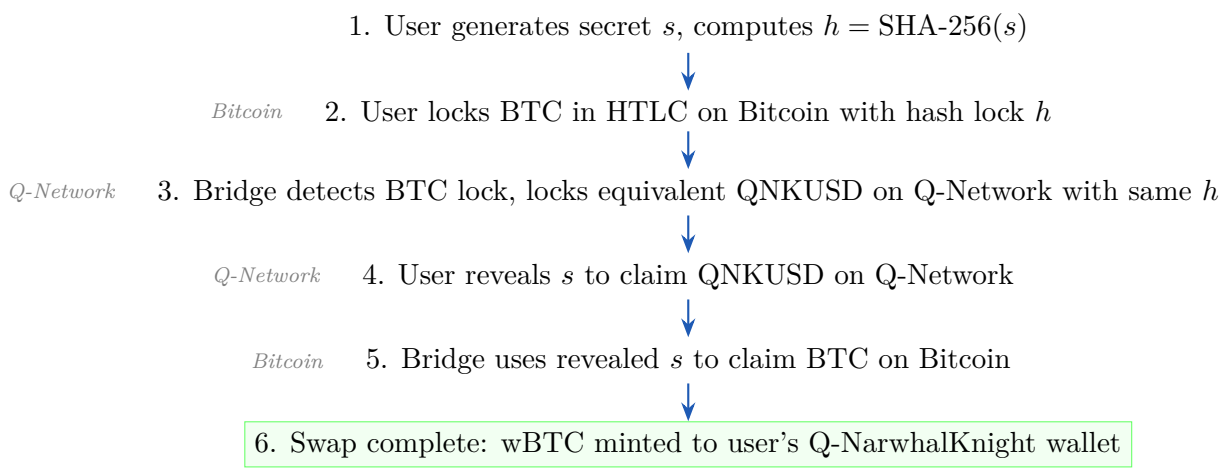

Figure 1: Atomic swap protocol sequence for BTC → wBTC bridge deposit.

3.6 Swap State Machine

Each atomic swap transitions through a deterministic state machine:

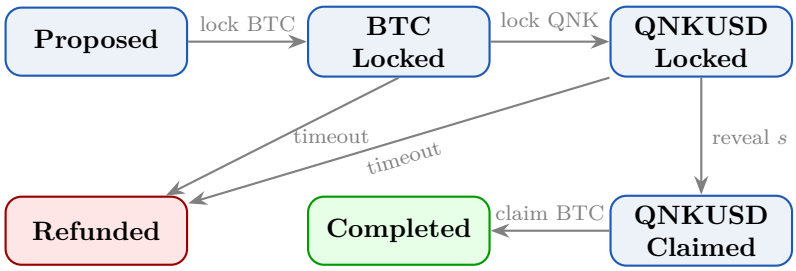

Figure 2: Atomic swap state machine.

4 Decentralized Multi-Signature Committee

Version 2.0 of the bridge protocol replaces single-node verification with a *rotating multi-signature committee* that independently validates every bridge claim before any mint or burn operation is executed. This eliminates the most significant trust assumption in Version 1.0: that a single bridge mediator honestly verifies deposits.

4.1 Committee Architecture

The committee consists of $N = 11$ validator nodes selected from the active peer set, with a Byzantine fault tolerance threshold of $t = 7$ (a supermajority of $\lceil \frac{2N}{3} \rceil + 1$). This ensures the system tolerates up to 4 malicious or offline committee members while maintaining liveness.

Table 3: Committee Parameters

Parameter	Value	Rationale
Committee Size (N)	11	Sufficient diversity; manageable gossip overhead
Threshold (t)	7	$\lceil 2 \times 11/3 \rceil + 1$; tolerates 4 Byzantine faults
Epoch Length	100 blocks	~100 seconds; frequent rotation prevents collusio
Attestation Timeout	60 seconds	Bounded latency; stale claims auto-expire
Gossipsub Topic	`/qnk/.../bridge-attestations`	Dedicated P2P channel

4.2 Deterministic Committee Selection

Committee membership is computed deterministically from on-chain state, ensuring all honest nodes agree on the current committee without coordination:

$$\text{score}(p) = \text{SHA3-256}(\texttt{block\_hash} \| \text{epoch} \| \texttt{peer\_id}(p)) \tag{2}$$

For a given epoch $e = \lfloor \texttt{block\_height}/100 \rfloor$, the committee is the set of N peers with the *lowest* score values. The use of the latest block hash as entropy prevents pre-computation of future committees.

Algorithm 1 Committee Selection

Require: Active peer set P, current block hash B_h, epoch e
Ensure: Committee $C \subseteq P$ with $|C| = \min(N, |P|)$

```
scored ← []
for all p ∈ P do
    s ← SHA3-256(B_h ‖ e ‖ peer_id(p))
    scored.push((s, p))
end for
scored.sort_by_key(|(s, _)| s)
return scored[0..N].map(|(_, p)| p)
```

4.3 Attestation Protocol

When a user claims a bridge swap by revealing the HTLC secret s, the following multi-sig verification protocol executes:

1. User submits claim with secret s to local node
2. Local node broadcasts `AttestationRequest` to committee via gossipsub
3. Each committee member independently verifies $H(s) = \texttt{hash\_lock}$
4. Valid: member signs attestation with Ed25519 and broadcasts back
5. Originator collects attestations; waits for ≥ 7 approvals
6. Threshold met: mint/burn executes; claim result returned to user

Figure 3: Multi-sig bridge attestation protocol for claim verification.

4.4 CBOR Message Format

Attestation messages are serialized using CBOR (Concise Binary Object Representation, RFC 8949) for compact binary encoding over gossipsub:

Attestation Message Types (CBOR)

```
// Request (broadcast by claiming node)
AttestationRequest {
    claim_id: String,          // Unique claim identifier
    chain: BridgeChainId,      // Bitcoin | Zcash | IronFish | Ethereum
    wallet: [u8; 32],          // Claimer's wallet address
    hash_lock: [u8; 32],       // Expected SHA-256 hash
    secret: [u8; 32],          // Revealed preimage
    amount: u128,              // Claimed amount
    swap_id: String,           // Atomic swap reference
    peer_id: String,           // Requesting peer
    signature: Vec<u8>,        // Ed25519 signature over fields
}

// Response (from each committee member)
Attestation {
    claim_id: String,          // Matching claim ID
    approved: bool,            // Verification result
    peer_id: String,           // Attesting peer
    signature: Vec<u8>,        // Ed25519 signature
    reason: Option<String>,    // Rejection reason if !approved
}
```

4.5 Committee Rotation and Epoch Transitions

Every 100 blocks, the active committee rotates. The transition protocol ensures smooth handoff:

1. At block h where $h \bmod 100 = 0$, a new epoch begins.
2. The committee is recomputed using the block hash at height h.
3. Pending claims from the previous epoch are *not* interrupted—they continue with the committee that initiated them until resolution or timeout.
4. Expired pending claims (older than 60 seconds) are garbage-collected every 30 seconds.

This rotation frequency of ∼100 seconds means that even if an adversary compromises committee members, they must re-compromise a new set of members every epoch. With P total peers, the probability that an adversary controlling f peers captures $\geq t$ committee seats is:

$$\Pr[\text{capture}] = \sum_{k=t}^{N} \binom{N}{k} \left(\frac{f}{P}\right)^{k} \left(1 - \frac{f}{P}\right)^{N-k} \tag{3}$$

For $P = 50$ peers, $f = 5$ malicious, $N = 11$, $t = 7$: $\Pr[\text{capture}] < 10^{-6}$ per epoch.

4.6 Graceful Degradation

When fewer than $t = 7$ peers are available (e.g., during network bootstrap or partitions), the bridge falls back to single-node validation with the local node as sole verifier. This ensures liveness at the cost of temporarily reduced decentralization, with a prominent log warning:

```
WARN: Bridge committee has only 3/7 required members.
      Falling back to single-node verification.
```

Once the peer count recovers above the threshold, multi-sig validation automatically resumes.

5 Wrapped Bitcoin (wBTC)

5.1 Overview

Wrapped Bitcoin enables Bitcoin holders to participate in Q-NarwhalKnight's DeFi ecosystem without selling their BTC on centralized exchanges. Each wBTC token is backed 1:1 by BTC locked in a P2WSH HTLC on the Bitcoin mainnet.

5.2 Deposit Flow (BTC → wBTC)

1. User initiates a `sell_btc` swap via the Q-NarwhalKnight bridge API.
2. The system generates a P2WSH HTLC address with the user's hash lock.
3. User sends BTC to the HTLC address on Bitcoin.
4. Upon confirmation (≥ 1 block), the bridge locks equivalent QNKUSD on Q-Network.
5. User reveals the secret preimage to claim QNKUSD.
6. The `mint_wrapped_token` function credits wBTC to the user's Q-NarwhalKnight wallet.
7. Bridge uses the revealed secret to claim the locked BTC.

5.3 Withdrawal Flow (wBTC → BTC)

1. User initiates a `buy_btc` swap.
2. The system verifies the user holds sufficient wBTC.
3. An HTLC is created on both chains with a new hash lock.
4. Upon successful claim, `burn_wrapped_token` removes wBTC from the user's balance.
5. The equivalent BTC is released from the HTLC on Bitcoin.

5.4 Bitcoin-Specific Security

- **Confirmation Depth:** Minimum 1 confirmation required before minting (configurable via `min_confirmation_depth`).
- **Segregated Witness:** All HTLCs use P2WSH for reduced fees and malleability resistance.
- **Fee Estimation:** Dynamic fee estimation ensures timely confirmation on the Bitcoin network.

6 Wrapped Zcash (wZEC)

6.1 Overview

Zcash's zero-knowledge proofs (zk-SNARKs) provide the strongest on-chain privacy guarantees of any major cryptocurrency. The wZEC bridge preserves these privacy properties during the cross-chain process by supporting both transparent and shielded transactions.

6.2 Privacy-Preserving Bridge Architecture

The Zcash bridge supports two modes of operation:

Transparent Mode: Standard HTLC atomic swap using t-addresses. Functionally identical to the Bitcoin bridge but on the Zcash network.

Shielded Mode: Uses z-addresses (Sapling) to hide the sender, receiver, and amount of the Zcash-side transaction. The HTLC hash lock is embedded in the memo field of a shielded transaction.

6.3 Shielded Address Management

Each Q-NarwhalKnight wallet can associate a Zcash shielded address:

$$\texttt{zcash\_z\_address} : \texttt{wallet\_address} \longrightarrow \texttt{z\_address} \tag{4}$$

This mapping is stored in RocksDB under the `CF_MANIFEST` column family and persists across restarts.

6.4 Zcash-Specific Parameters

- **Timelock:** 120 blocks ($\approx$ 2.5 hours at 75 seconds/block)
- **Confirmation Depth:** 10 confirmations for shielded, 1 for transparent
- **zk-SNARK Verification:** Zcash's Sapling proofs are verified on-chain before mint

7 Wrapped Iron Fish (wIRON)

7.1 Overview

Iron Fish is an encrypted-by-default Layer 1 blockchain where every transaction is private. Unlike Zcash's optional shielding, Iron Fish enforces privacy at the protocol level, making it the most privacy-focused bridge target in our system.

7.2 Encrypted Bridge Integration

The Iron Fish bridge interfaces with Iron Fish's RPC API to create and manage atomic swaps:

1. **Account Management:** Bridge manages Iron Fish accounts via `wallet/create` RPC.
2. **Transaction Creation:** Uses `wallet/sendTransaction` with memo field for hash lock embedding.
3. **Balance Verification:** Queries `wallet/getBalance` to verify deposits before minting.

7.3 Iron Fish-Specific Parameters

- **Timelock:** 60 blocks ($\approx$ 60 minutes at 60 seconds/block)
- **Confirmation Depth:** 2 confirmations (Iron Fish's faster block time allows fewer)
- **Privacy Guarantee:** All Iron Fish transactions are encrypted; the bridge never reveals transaction amounts to observers.

8 Wrapped Ethereum (wETH)

8.1 Overview

Ethereum is the largest smart contract platform by total value locked, hosting the majority of DeFi protocols, NFT marketplaces, and Layer 2 scaling solutions. The wETH bridge brings Ethereum's liquidity into Q-NarwhalKnight's quantum-resistant ecosystem, while providing Ethereum users access to QNK's sub-3-second finality and post-quantum protections.

8.2 Solidity HTLC Contract

Unlike Bitcoin's script-based HTLCs, Ethereum bridge HTLCs are implemented as a Solidity smart contract deployed on the Ethereum mainnet:

Ethereum HTLC Contract (Simplified)

```
contract QNKBridgeHTLC {
    struct Swap {
        address sender;
        bytes32 hashLock;
        uint256 amount;
        uint256 timelock;
        bool claimed;
        bool refunded;
    }

    function newSwap(bytes32 _hashLock, uint256 _timelock)
        external payable returns (bytes32 swapId);

    function claim(bytes32 _swapId, bytes32 _secret)
        external;  // requires SHA256(secret) == hashLock

    function refund(bytes32 _swapId)
        external;  // requires block.timestamp >= timelock
}
```

8.3 Deposit Flow (ETH → wETH)

1. User initiates a bridge deposit via the Q-NarwhalKnight API.
2. The system generates a hash lock and deploys (or calls) the HTLC contract on Ethereum.
3. User sends ETH to the HTLC contract with the hash lock.
4. Upon sufficient confirmations ($\geq$ 12 blocks), the bridge committee validates the deposit.
5. With ≥ 7 attestations confirming the deposit, wETH is minted to the user's Q-NarwhalKnight wallet.
6. The user reveals the secret to claim; the bridge claims the locked ETH.

8.4 Withdrawal Flow (wETH → ETH)

1. User initiates a withdrawal and specifies their Ethereum address.
2. The bridge committee verifies the user holds sufficient wETH.
3. Upon 7-of-11 committee approval, wETH is burned and the HTLC on Ethereum is funded.
4. User claims ETH from the HTLC by revealing the secret.

8.5 Ethereum-Specific Considerations

- **Gas Optimization:** HTLC contract uses minimal storage and avoids loops to reduce gas costs. Estimated gas: 65,000 for `newSwap`, 35,000 for `claim`.
- **EIP-1559 Fee Handling:** Dynamic base fee with priority fee estimation for timely inclusion.
- **18-Decimal Precision:** ETH uses 18 decimals (Wei), requiring careful conversion. Internally, wETH balances are stored in Wei precision.
- **Confirmation Depth:** 12 confirmations required ($\approx$ 2.4 minutes at 12 seconds/block) to mitigate reorganization risk.
- **Smart Contract Verification:** The HTLC contract source code is verified on Etherscan for transparency.

9 DEX Integration

9.1 First-Class DEX Citizenship

Wrapped bridge tokens are treated as first-class tokens on Q-NarwhalKnight's built-in decentralized exchange. They appear alongside native tokens (QUG, QUGUSD) and user-deployed custom tokens in the supported tokens list.

Table 4: DEX Trading Pairs Enabled by Bridge Tokens

Pair	Description	Use Case
wBTC / QUG	Bitcoin–QUG exchange	BTC holders enter QNK ecosystem
wBTC / QUGUSD	Bitcoin–stablecoin	BTC hedging with stablecoin
wZEC / QUG	Zcash–QUG exchange	Privacy coin diversification
wZEC / QUGUSD	Zcash–stablecoin	Private stablecoin acquisition
wIRON / QUG	Iron Fish–QUG exchange	Encrypted asset trading
wIRON / QUGUSD	Iron Fish–stablecoin	Private stablecoin acquisition
wETH / QUG	Ethereum–QUG exchange	ETH DeFi users enter QNK
wETH / QUGUSD	Ethereum–stablecoin	ETH hedging with stablecoin

9.2 Automated Market Maker (AMM)

All bridge token pairs use Q-NarwhalKnight's constant-product AMM ($x \cdot y = k$) with the following properties:

- **24-Decimal Reserve Format:** All reserves are stored in 24-decimal precision internally, regardless of the token's native decimals (8 for bridge tokens).
- **0.3% Swap Fee:** Standard fee applied to all swaps, distributed to liquidity providers.
- **Configurable Slippage:** Users set their own slippage tolerance (default 0.5%).
- **Real-Time Price Discovery:** Server-Sent Events (SSE) emit price updates after every swap.

9.3 Liquidity Pool Creation

When a bridge token is first minted, liquidity pools can be created by any user:

Pool Creation API

```
POST /api/v1/dex/pool/create
{
    "tokenA": "wBTC",
    "tokenB": "QUG",
    "amountA": "100000000",      // 1 wBTC (8 decimals)
    "amountB": "42500000000..."  // ~$42.50 worth of QUG (24 decimals)
}
```

9.4 Price Oracle Integration

Bridge token prices are available through Q-NarwhalKnight's oracle system:

- `/v1/oracle/price/wBTC` — Current wBTC price in USD (derived from pool reserves)
- `/v1/oracle/price/wZEC` — Current wZEC price
- `/v1/oracle/price/wIRON` — Current wIRON price
- Price history available for charting via the price history indexer

10 Privacy and Censorship Resistance

10.1 Tor Integration

All bridge communications are routed through Tor, preventing IP address correlation between bridge participants:

- Bitcoin RPC connections proxied through `127.0.0.1:9050` (Tor SOCKS5)
- Q-NarwhalKnight P2P connections use Tor-compatible libp2p transports
- No direct IP-to-IP connections between bridge participants

10.2 Steganographic Node Discovery

Rather than using a centralized registry, Q-NarwhalKnight bridge nodes discover each other through steganographic advertisements embedded in Bitcoin transactions. Four encoding methods are employed:

Distributed Encoding: Node advertisements are split into 16-byte fragments spread across multiple Bitcoin transactions, with sequence IDs, SHA3-256 checksums, and random padding.

Value Pattern Encoding: Data bits are encoded in Bitcoin output values—even satoshi values represent 0, odd values represent 1—appearing as normal change outputs to observers.

Timing Pattern Encoding: Information is encoded in the delays (60–300 seconds) between related transactions, with timing hints for reconstruction.

Address Pattern Encoding: Specific change address combinations encode node information through pattern selection.

This approach uses Bitcoin's immutable ledger as a censorship-resistant bulletin board: as long as Bitcoin transactions are not censored, Q-NarwhalKnight bridge nodes can discover each other.

10.3 Cover Traffic

When steganographic discovery is enabled, the bridge generates cover traffic—additional Bitcoin transactions that carry no real data but are indistinguishable from advertisement fragments. This prevents statistical analysis from identifying which transactions contain node discovery data.

11 Security Analysis

11.1 Threat Model

We consider the following adversary capabilities:

1. **Network-Level:** Can observe and delay (but not indefinitely block) messages between parties.
2. **Computational:** Bounded by classical or quantum computational resources.
3. **Economic:** May attempt to profit by manipulating the bridge protocol.

11.2 Security Properties

Atomicity: The HTLC construction ensures that revealing the secret s on one chain immediately enables claiming on the other. An adversary cannot claim on one chain while preventing the counterparty from claiming on the other, because the secret is publicly visible once used.

No Double-Spend: A user cannot both claim the bridged assets and refund the locked native assets. The claim path requires revealing s (enabling counterparty claim), while the refund path requires waiting for the timelock (by which time the counterparty has already claimed).

Time-Bounded Exposure: Locked funds are never permanently frozen. The staggered timelock architecture ensures that if the protocol stalls at any point, all parties can recover their funds within 24 hours (Bitcoin) or sooner (Zcash: 2.5 hours, Iron Fish: 1 hour, Ethereum: 2.4 minutes).

Multi-Sig Byzantine Tolerance: The 7-of-11 committee threshold tolerates up to 4 malicious or offline validators. Committee rotation every 100 blocks ($\sim$100 seconds) prevents sustained collusion. Deterministic selection via SHA3 ensures all honest nodes independently agree on committee membership without coordination messages.

Post-Quantum Safety: Once bridged, wrapped tokens inherit Q-NarwhalKnight's quantum-resistant protections. The SHA-256 hash lock provides 128-bit post-quantum security (via Grover's algorithm), which remains sufficient for the short-lived HTLC duration. Attestation signatures use Ed25519 with a planned migration path to Dilithium5 post-quantum signatures.

No Single Point of Failure: Unlike centralized bridge designs (e.g., wrapped BTC on Ethereum which relies on BitGo custodians), the Q-NarwhalKnight bridge requires independent verification by a supermajority of rotating committee members. No single node can unilaterally approve a mint or burn.

11.3 Invariants

The system maintains three critical invariants:

$$\sum_{\text{all wallets}} \texttt{wBTC\_balance} = \sum_{\text{all HTLCs}} \texttt{locked\_BTC} \tag{5}$$

$$\forall\, \texttt{mint}(w, a) : \ \exists\, \texttt{HTLC\_lock}(a) \text{ with } \geq 1 \text{ confirmation} \tag{6}$$

$$\forall\, \texttt{burn}(w, a) : \ \texttt{balance}(w) \geq a \tag{7}$$

Equation 3 ensures total supply backing. Equation 4 prevents unbacked minting. Equation 5 prevents over-burning.

12 Audit and Persistence

12.1 Operation Logging

Every bridge operation (mint or burn) is recorded as a `BridgeOperation` in RocksDB:

Table 5: Bridge Operation Record Fields

Field	Type	Description
`op_id`	String	Unique identifier: `bridge_{chain}_{swap_id}`
`chain`	Enum	Bitcoin, Zcash, or IronFish
`op_type`	Enum	Mint (deposit) or Burn (withdrawal)
`wallet`	[u8; 32]	User's Q-NarwhalKnight wallet address
`amount`	u128	Amount in native base units (satoshis/zatoshis/ore)
`native_txid`	Option⟨String⟩	Transaction ID on the native chain
`swap_id`	Option⟨String⟩	Associated atomic swap identifier
`timestamp`	DateTime	UTC timestamp of the operation
`status`	Enum	Pending, Confirmed, or Failed(reason)

12.2 Storage Schema

Bridge data uses Q-NarwhalKnight's RocksDB column family `CF_MANIFEST`:

Storage Key Format

```
bridge_op:{op_id}                    -> Serialized BridgeOperation (JSON)
bridge_op_idx:{wallet_hex}:{op_id}   -> op_id (wallet index)
atomic_swap:{swap_id}                -> Serialized AtomicSwapProposal (JSON)
atomic_swap_idx:{wallet}:{swap_id}   -> swap_id (wallet index)
btc_swap_dir:{swap_id}               -> "buy_btc" | "sell_btc"
zcash_z_address:{wallet}             -> z_address (shielded)
zcash_balance:{wallet}               -> zatoshi balance
ironfish_address:{wallet}            -> Iron Fish address
ironfish_balance:{wallet}            -> ore balance
```

12.3 Recovery Procedures

On node restart, the bridge system recovers its state:

1. **Token Balances:** Loaded from RocksDB's `CF_TOKEN_BALANCES` into the in-memory HashMap.

2. **Pending Swaps:** All active atomic swaps are loaded via `load_all_atomic_swaps()` and their HTLC states are re-checked against the native chain.

3. **Bridge Operations:** Audit trail is preserved and queryable via `list_bridge_operations()`.

13 Lessons Learned

The bridge protocol's design incorporates hard-won lessons from Q-NarwhalKnight's production history:

1. **Immediate Persistence for Critical Operations:** Token balance persistence was initially batch-synced every 15 seconds. A server crash within that window could lose bridge mints. Bridge operations now trigger immediate `save_token_balance()` calls.

2. **Decimal-Aware Thresholds:** An early sanity check (`MAX_SANE_BALANCE = 1e31`) would silently reset legitimate large token balances to zero. All thresholds are now decimal-aware: $10^{(\text{decimals}+18)}$.

3. **Contract Metadata for Decimals:** Pool-stored decimals proved unreliable (defaulting to 24 for auto-created pools). The bridge always reads decimals from contract metadata, never from pool configuration.

4. **Canonical Token Format:** P2P-received pool data stored tokens in pure hex format, while local pools used symbols. The bridge canonicalizes all token references to prevent lookup failures.

5. **Multi-Sig over Single-Node:** Version 1.0 relied on a single bridge mediator for deposit verification. While the HTLC protocol itself is trustless, the *triggering* of mint operations was a centralization risk. Version 2.0's rotating committee ensures that no single compromised node can unilaterally mint wrapped tokens.

6. **Graceful Degradation:** Hard requirements for committee quorum would make the bridge unavailable during network bootstrap or partitions. The fallback to single-node validation preserves liveness while clearly logging the reduced security posture.

7. **CBOR over JSON for Gossipsub:** Binary CBOR encoding reduces attestation message size by ∼40% compared to JSON, reducing gossipsub bandwidth consumption in the critical path.

14 API Reference

14.1 Bitcoin Bridge Endpoints

Method	Endpoint	Description
`POST`	`/api/v1/bitcoin/swap`	Create new atomic swap
`GET`	`/api/v1/bitcoin/swap/:id`	Get swap status
`POST`	`/api/v1/bitcoin/swap/:id/claim`	Claim swap with secret
`POST`	`/api/v1/bitcoin/swap/:id/refund`	Refund expired swap
`GET`	`/api/v1/bitcoin/swaps`	List user's swaps
`GET`	`/api/v1/bitcoin/balance`	Get BTC bridge balance

14.2 Zcash Bridge Endpoints

Method	Endpoint	Description
POST	/api/v1/zcash/swap	Create shielded swap
GET	/api/v1/zcash/swap/:id	Get swap status
POST	/api/v1/zcash/swap/:id/claim	Claim with secret
POST	/api/v1/zcash/swap/:id/refund	Refund expired swap
GET	/api/v1/zcash/swaps	List user's Zcash swaps

14.3 Iron Fish Bridge Endpoints

Method	Endpoint	Description
POST	/api/v1/ironfish/swap	Create encrypted swap
GET	/api/v1/ironfish/swap/:id	Get swap status
POST	/api/v1/ironfish/swap/:id/claim	Claim with secret
POST	/api/v1/ironfish/swap/:id/refund	Refund expired swap
GET	/api/v1/ironfish/swaps	List user's Iron Fish swaps

14.4 Ethereum Bridge Endpoints

Method	Endpoint	Description
POST	/api/v1/ethereum/swap	Create ETH atomic swap
GET	/api/v1/ethereum/swap/:id	Get swap status
POST	/api/v1/ethereum/swap/:id/claim	Claim swap with secret
POST	/api/v1/ethereum/swap/:id/refund	Refund expired swap
GET	/api/v1/ethereum/swaps	List user's ETH swaps
GET	/api/v1/ethereum/balance	Get wETH bridge balance
GET	/api/v1/ethereum/gas	Current gas price estimate

14.5 Bridge Committee Endpoints

Method	Endpoint	Description
GET	/api/v1/bridge/committee	Current committee members
GET	/api/v1/bridge/committee/epoch	Current epoch and rotation info
GET	/api/v1/bridge/attestations/:claim_id	Attestation status for a claim
GET	/api/v1/bridge/stats	Bridge statistics and metrics

All endpoints require authentication via `X-Wallet-Auth` header.

15 Conclusion

Version 2.0 of the Q-NarwhalKnight Cross-Chain Bridge Protocol advances the state of the art in trustless cross-chain interoperability by combining Hash Time-Locked Contracts with a **decentralized 7-of-11 multi-signature committee**. This architecture eliminates the single-point-of-failure risk present in centralized bridge designs while maintaining the atomic swap guarantees that prevent partial execution or fund loss.

The addition of **Ethereum** (wETH) alongside Bitcoin (wBTC), Zcash (wZEC), and Iron Fish (wIRON) expands the bridge to cover four major blockchain ecosystems, representing the majority of crypto market capitalization. All wrapped tokens are first-class citizens on Q-NarwhalKnight's DEX with sub-3-second finality through DAG-Knight consensus.

Key innovations in Version 2.0 include:

- **Rotating Committee:** Deterministic SHA3-based selection with 100-block epoch rotation prevents sustained collusion.
- **Byzantine Fault Tolerance:** 7-of-11 threshold tolerates up to 4 malicious or offline validators.
- **Graceful Degradation:** Automatic fallback to single-node verification when committee quorum is unavailable.
- **CBOR Gossipsub:** Efficient binary attestation messages over a dedicated P2P topic.
- **Ethereum Smart Contract HTLCs:** Solidity-based HTLC with gas-optimized claim and refund paths.

As the bridge matures, we anticipate implementing threshold signature schemes (FROST) for committee signing, adding ERC-20 token bridge support, and extending to Layer 2 networks (Arbitrum, Optimism) for reduced gas costs.

Q-NarwhalKnight Foundation

Building the quantum-resistant financial infrastructure of tomorrow.

`https://quillon.xyz`