qBitcoin (QBTC): Evolving Bitcoin from Proof-of-Work Distribution to Post-Quantum Proof-of-Stake Stewardship

qBitcoin Core Contributors

September 25, 2026

Abstract

qBitcoin (QBTC) transitions the Bitcoin network from Proof-of-Work (PoW) distribution to Proof-of-Stake (PoS) stewardship. Recognizing the functional completion of the initial supply issuance, the protocol preserves the stateless UTXO model and the distribution established under Proof-of-Work, admitting up to half of the 21 million base through a pegged bidirectional migration bridge whose conversion rate falls as upgraded volume grows, so that QBTC upgraded early can later be exchanged for more BTC. The bridged supply is thereby capped at $\sim$3.9 Million QBTC, and issuance beyond it is confined to a terminal emission schedule, capped at 4 Million QBTC and activated when the bridge closes, which funds validators and network liveness rather than expanding the money supply. Total supply is therefore bounded at $\sim$7.9 Million QBTC, with no perpetual inflation.

To proactively mitigate emergent security threats, the network integrates lattice-based post-quantum cryptography (PQC) alongside classical Schnorr and ECDSA signatures. By replacing computational expenditure with UTXO-based staking, the architecture achieves rapid transaction finality and long-term sustainability while enabling native yield generation, allowing asset holders to secure the ledger and earn predictable rewards without freezing or locking capital.

Contents

1 Introduction: The Maturity of Decentralized Money **4**

2 Background and Problem Statement: The Limits of Legacy Architecture **5**
2.1 Consensus and Issuance: The Limits of Proof-of-Work 5
2.2 Transaction Security and the Quantum Threat 5
2.3 Protocol Upgrade Friction 6

3 System Overview: The qBitcoin Architecture **6**

4 The Migration and Economic Bootstrapping **6**
4.1 The Forward Upgrade: From Issuance to Stewardship 9
4.1.1 The Conversion Rate Schedule 9
4.1.2 Upgrade Stop Conditions 11
4.2 The Network Reward Fund: Consolidated Economics & Subsidies 13
4.2.1 A Single Pot and a Single Payout 13
4.2.2 The Fund as an Exponential Moving Average 15
4.2.3 Activity-Scaled Distribution 16
4.2.4 The Terminal Emission Schedule 16
4.2.5 Soft Fee Priority 18
4.2.6 Supply Accounting and the Reserve Curve 19
4.3 The Trust-Minimized Downgrade Protocol & Reclaim Guardrails 20
4.3.1 Architectural Necessity of the Conversion Service 20
4.3.2 The Downgrade State Machine 21
4.3.3 The Reclaim Path: Fallback Guardrails 21

5 Network Stewardship and Consensus: UTXO-based Proof-of-Stake **22**
5.1 Block Generation and Stake Weight 22
5.2 The Genesis Stake 24
5.3 Delegated Staking 24
5.4 Network Health and Transaction Prioritization 25
5.5 Network Security: Fork Choice and the Reorganization Penalty 26
5.6 Slashing: Penalizing Equivocation 28
5.6.1 Equivocation and Its Proof 28
5.6.2 The Slashing Transaction 29
5.6.3 Branch Invalidation 29
5.6.4 Weight Accounting and Inclusion Incentive 29
5.6.5 Scope and Limits 30

6 Cryptographic and Architectural Upgrades **30**
6.1 Simplified Address System (P2SH-Only) 30
6.2 Post-Quantum and Crypto-Agile Signatures 31
6.3 Enhanced Script Privacy with MAST 31
6.4 Client Implementation and Interoperability 32

7 Extended Capabilities: Client-Side Smart Contracts **33**
7.1 Design Philosophy: A Hybrid Validation Model 33
7.2 UTXO-based Implementation Concept 34
7.3 Architectural References and Design Lineage 34

8 Roadmap: The Evolution Path **35**
8.1 Phase 1: The Distribution (2009-2025) 35
8.2 Phase 2: The Evolution (2025-2026) 35
8.3 Phase 3: The Transition (2026-2027) 36
8.4 Phase 4: The Utility and Stewardship (2027+) 36

9 Conclusion **37**

Disclaimer **37**

1 Introduction: The Maturity of Decentralized Money

Bitcoin [1] established two fundamental mechanisms: decentralized consensus for verifying transactions and a Proof-of-Work (PoW) algorithm for decentralized supply distribution. For nearly two decades, the Proof-of-Work mechanism has managed the initial issuance of the fixed supply.

With the vast majority of that supply having entered circulation as self-custodied Unspent Transaction Outputs (UTXOs), the distribution phase is functionally complete. Consequently, the network's primary technical and economic priority must shift from issuance to long-term stewardship. The continued reliance on energy-intensive computation, a tool designed for distribution, is no longer an efficient nor sustainable method for securing the ledger indefinitely.

The present document introduces qBitcoin (QBTC), a comprehensive protocol upgrade designed to transition Bitcoin to the new era of stewardship. qBitcoin retires the PoW algorithm in favor of a UTXO-based, post-quantum Proof-of-Stake (PoS) [2] consensus layer. The PoS transition decouples the network's security from energy consumption while significantly increasing transaction throughput. The proposed consensus model introduces a compelling economic motivation for asset holders: assets naturally generate predictable yield directly from the network, paid out of protocol income rather than perpetual issuance, and without requiring capital to be locked.

To facilitate a smooth and trust-minimized transition, the protocol employs a pegged bidirectional migration bridge managed by a legally accountable Conversion Service. Asset holders upgrade legacy Bitcoin to qBitcoin by providing on-chain proof of lock, and may reversibly downgrade back to legacy reserves via cryptographic proof of burn. Custodial counterparty risk is bounded rather than eliminated: the legacy reserve sits in a multisignature address held by the Conversion Service federation, and no qBitcoin rule can spend from it. What the protocol does enforce lies on its own side of the bridge, where every downgrade in flight rests behind a timelocked reclaim branch, so a holder whose conversion stalls recovers their QBTC in full without anyone's cooperation. The conversion rate starts at one QBTC per BTC and falls as upgraded volume grows. Later holders therefore receive fewer QBTC for each BTC they lock, the unconverted BTC stays in the reserve behind the coins already in circulation, and QBTC upgraded early can later be exchanged for more BTC than was locked for it, which rewards early adoption. A flat 1% of every mint capitalizes a permanent endowment, the Network Reward Fund, which is the sole source of validator compensation. The bridge itself admits at most half of the legacy supply and is open for at most 10 years, closing early if a designated set of the earliest legacy outputs is ever spent; when it closes, a terminal emission schedule succeeds the conversion fee as the network's non-fee income, minting a halving per-timeslot amount capped in total at 4 Million QBTC and paying it directly to block proposers. Issuance is therefore bounded and terminal in both of its phases rather than perpetually inflationary. Furthermore, the upgraded architecture leverages the efficiency of the UTXO model to support a hybrid smart contract ecosystem, balancing lightweight native assets with scalable, off-chain client execution for complex applications.

In essence, qBitcoin preserves Bitcoin's foundational principles of a bounded, terminal supply schedule, a verifiable ledger history, and the stateless UTXO model, mapping legacy locked coins to the new post-quantum UTXOs with PoS consensus layer. As a result, the presented protocol upgrades the network with high throughput, post-quantum-ready security, long-term economic viability, and native, liquid yield generation for asset holders.

The remainder of the paper is structured as follows: Section 2 details the limitations of the legacy architecture, including economic vulnerabilities and the quantum threat. Section 3

provides a high-level overview of the qBitcoin system architecture. Section 4 explains the dynamic migration mechanism and the economic bootstrapping of the network. Section 5 defines the UTXO-based Proof-of-Stake consensus model, the fork choice rules, and the slashing mechanism that penalizes equivocation. Section 6 outlines the core cryptographic upgrades, including the integration of post-quantum signatures and enhanced script privacy. Section 7 introduces the hybrid validation model for client-side smart contracts, and Section 8 presents the evolutionary roadmap of the protocol. Finally, Section 9 concludes the paper.

2 Background and Problem Statement: The Limits of Legacy Architecture

The architecture of qBitcoin is an iterative advancement of the Bitcoin protocol, designed to preserve foundational scarcity while modernizing security and consensus primitives. An evaluation of the legacy system reveals a fundamental misalignment between its original distribution-focused design and the current requirements of network stewardship.

2.1 Consensus and Issuance: The Limits of Proof-of-Work

The legacy Bitcoin protocol utilizes a SHA-256 Proof-of-Work (PoW) algorithm to achieve both network consensus and decentralized supply distribution [1]. Under the PoW model, network participants (miners) compete to solve cryptographic puzzles to earn the right to append a block of transactions to the ledger, receiving newly minted coins as a reward. As the economic value of the network has grown, the resulting competitive incentives have driven the aggregate computational expenditure required to solve these puzzles to globally significant levels.

While the PoW incentive structure was effective for initial supply issuance and bootstrapping network security, its continued use in a mature, high-demand state presents several structural challenges:

- **Architectural Latency and Inefficiency:** The ten-minute block interval, required for PoW's probabilistic security, imposes a high-latency ceiling on transaction finality [1]. With over 94% of the supply now in circulation, the energy consumption required to secure the network is an inefficient operational cost for a system primarily focused on asset stewardship.
- **Economic Vulnerability:** The legacy security model relies on a diminishing block subsidy that must eventually transition to a pure fee-only market. The diminishing subsidy creates long-term uncertainty regarding the ability of a fee-only market to consistently finance the energy expenditure required to protect the chain against 51% attacks in perpetuity [1].

2.2 Transaction Security and the Quantum Threat

The security of asset ownership on the legacy ledger depends on the Elliptic Curve Digital Signature Algorithm (ECDSA) using the `secp256k1` curve [3]. While robust against classical adversaries, ECDSA is vulnerable to the advancement of quantum computing.

Specifically, the protocol's reliance on ECDSA bases its security on the Elliptic Curve Discrete Logarithm Problem (ECDLP). The best known classical algorithms require exponential

$\mathcal{O}(2^{k/2})$ time to solve this problem for a key size of k bits. However, given a sufficiently powerful quantum computer, Shor's algorithm solves the ECDLP in polynomial $\mathcal{O}(k^3)$ time, allowing an adversary to efficiently derive private keys from exposed public keys [4, 5]. The standardization of post-quantum signature schemes like Falcon proactively secures cryptographic primitives against Shor's algorithm to maintain the integrity of the supply schedule [6].

2.3 Protocol Upgrade Friction

The legacy protocol lacks a clear mechanism to address these issues. Historical friction regarding hard-forking upgrades [7] leaves the network without a viable path to implement necessary modernization, such as a transition to crypto-agile signature validation with post-quantum support.

These structural challenges drive the need for qBitcoin, a protocol upgrade that addresses performance, economic sustainability, and cryptographic security through a voluntary, bidirectional migration model that preserves two-way asset liquidity between ledgers.

3 System Overview: The qBitcoin Architecture

The qBitcoin protocol implements a comprehensive network upgrade that directly addresses the limitations of the legacy system, while preserving the coin distribution established by the legacy PoW Bitcoin blockchain. The architecture retains the foundational Bitcoin UTXO model while replacing the consensus mechanism to enhance performance, security, and economic sustainability.

The system replaces Proof-of-Work with a UTXO-based Proof-of-Stake (PoS) system. Network consensus is derived from the value and age (coin-age) of participant UTXOs, aligning the security model with the network's mature, post-distribution state. The UTXO-based coin-age PoS model allows for fluid, decentralized participation without the need for complex locking mechanisms, ensuring that assets preserve liquidity while actively contributing to network security, block issuance, and asset holders' yield through transaction fee collection.

To address emergent cryptographic threats while preserving interoperability, transactions are secured through a crypto-agile signature framework supporting Falcon alongside classical Schnorr and ECDSA. Falcon provides the post-quantum security path, while classical schemes remain available for compatibility and policy-defined transaction classes. Concurrently, the block interval is reduced to 10 seconds to support high-throughput applications.

Figure 1 provides a master overview of the qBitcoin architecture, illustrating how the network bridges from the legacy distribution phase into the new, layered stewardship stack. Table 1 summarizes these core architectural shifts. Table 2 consolidates the fixed protocol constants used throughout the remainder of the paper.

4 The Migration and Economic Bootstrapping

This section details the economic engine and bidirectional bridging mechanics of qBitcoin, structuring the transition from Proof-of-Work issuance to Proof-of-Stake stewardship across three core components:

- **The Forward Upgrade & Conversion Rate:** The mechanism for migrating legacy PoW BTC into PoS QBTC by locking it via the Conversion Service. The conversion rate, in

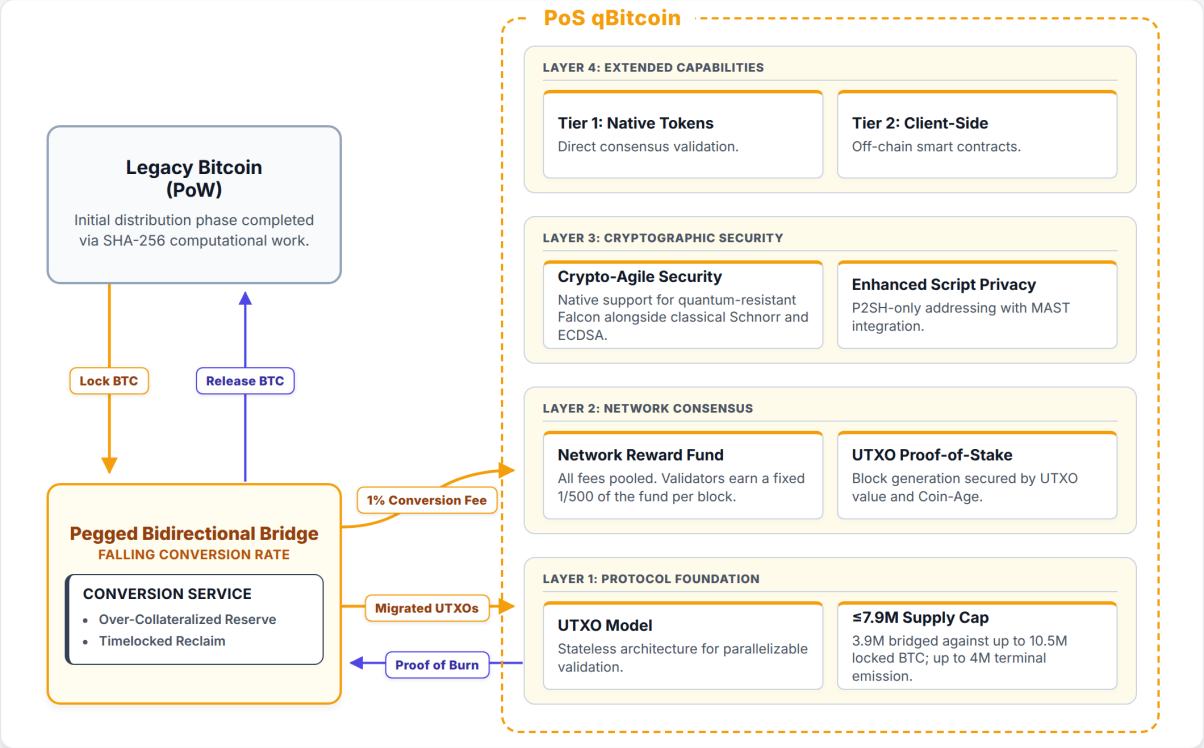


Figure 1: Overview of the qBitcoin system architecture. The protocol transitions from the legacy Proof-of-Work distribution phase (left) through a pegged bidirectional migration bridge, establishing a comprehensive Proof-of-Stake stewardship stack (right).

Table 1: Comparison of the Legacy Protocol and the qBitcoin Evolution.

Feature	**Legacy Bitcoin (BTC)**	**qBitcoin (QBTC)**
Consensus	Proof-of-Work (SHA-256)	UTXO Proof-of-Stake
Avg. Block Time	$\sim$10 minutes	10 seconds
Cryptography	ECDSA (Vulnerable)	Lattice-based PQC (Secure)
Total Supply	21 Million	$\leq$7.9M (3.9M bridged + 4M emitted)
Security Model	Energy Expenditure	Coin-Age + Slashing + Reward Fund
Smart Contracts	Limited Scripting	Tier 1 Native & Tier 2 Client-Side
Issuance	PoW block subsidy	Terminal emission (halving, capped)
Migration Path	N/A	Pegged Bidirectional Bridge
Migration Window	N/A	$\leq$10 years, cap, or veto; then closed
Reserve Backing	N/A	$\geq$1 BTC per bridged QBTC

QBTC minted per BTC locked, starts at 1:1 and falls as the volume already upgraded grows. Exactly 1% of every mint is credited to the Network Reward Fund, and nothing is minted for the part of a deposit that the lower rate leaves unconverted: that BTC stays in the reserve as collateral for the coins already in circulation. As a result, QBTC upgraded early can later be exchanged for more BTC than was locked for it, which rewards early adoption. The bridge admits at most half of the legacy supply, and it closes early if any of a designated set of legacy outputs is spent.

- **The Network Reward Fund & Subsidies:** An independent, permanent network en-

Table 2: Protocol constants of the qBitcoin network.

Parameter	**Symbol**	**Value**
Genesis Upgrade	—	2026-09-13 11:46:40 UTC
Genesis stake (service coins)	—	50 QBTC, stake-only
Target block interval (timeslot duration)	—	10 seconds
Heartbeat interval	—	100 timeslots (≈16.7 min)
Low-fee transaction allowance	—	1 per block
Minimum fee rate	—	dynamic, 10 sat/kb to 0.01 QBTC/kb
Reorganization propagation buffer	—	100 timeslots
Reorganization penalty ceiling	$P_{\max}$	32×
Penalty normalization horizon	—	$d_{\text{eff}} = 921{,}600$ timeslots (≈106 days)
Consensus checkpoints	—	cap reorganization depth
Upgrade level size	Δ	4,200 BTC upgraded per level
Conversion rate decay per level	δ	0.999 (5,000 levels)
Conversion fee (to Reward Fund)	—	1% of minted QBTC
Upgrade cap	$V_{\max}$	10,500,000 BTC standing in the reserve
Veto set	—	6 legacy outpoints (Table 4)
Conversion window	T_{conv}	$\leq$ 10 years from the Genesis Upgrade
Bridge close	T_{close}	first of T_{conv}, upgrade cap, veto
Emission activation	T_{close}	bridge close
Initial emission rate	s_0	0.2 QBTC per timeslot
Smallest indivisible unit	—	10^{-8} QBTC (1 satoshi)
Emission halving interval	H_{halve}	10,000,000 timeslots
Total terminal emission	—	$\leq$ 4 Million QBTC
Subsidy fraction	k	$1/500$ per block
Bridge transaction weight	A	value × one year of timeslots, per transaction
Stake output maturity (spends only)	M_{stake}	12 hours
Downgrade payout per QBTC	$1/m$	the conversion rate inverted, ≥ 1 BTC
Downgrade service fee	—	1%
Freeze reclaim timeout	—	48 hours
Downgrade-output reclaim timeout	—	7 days
Maximum redeem script	—	65,535 bytes
Maximum signature list	—	65,535 bytes
Slashing fine fraction	ϕ_{slash}	10%
Slashing evidence retention horizon	—	106 blocks

dowment capitalized by the 1% conversion fee, by the whole of every block's transaction fees, and by slashing fines. It is the sole pooled source of validator compensation, releasing a fixed fraction of its balance each block and thereby subsidizing mandatory heartbeat blocks for ongoing network stability. When the bridge closes and conversion fees stop arriving, a terminal emission schedule takes over as the network's non-fee income, minted directly to each eligible block's proposer rather than into the fund.

- **The Trust-Minimized Downgrade:** A protocol enabling the conversion of QBTC back into legacy BTC at the conversion rate of Section 4.1.1 read backwards, never below one BTC per QBTC, via an accountable Conversion Service, with a timelocked reclaim

branch at each step, so that a stalled conversion returns the holder's QBTC in full.

4.1 The Forward Upgrade: From Issuance to Stewardship

Throughout the conversion window, qBitcoin is minted exclusively against legacy Bitcoin locked within the migration bridge, and never at more than one QBTC per BTC. The bridge admits at most half of the legacy monetary base, 10,500,000 BTC, and it converts every locked coin at the rate the reserve volume has reached (Section 4.1.1), a rate that falls as the reserve fills and recovers as redemptions empty it. The bridged supply of qBitcoin therefore cannot exceed 3,855,669 QBTC at any moment, a figure fixed by the schedule alone and reached only if the upgrade cap is filled. Beyond the bridged supply the protocol issues one further, bounded quantity: after the bridge closes, the terminal emission schedule of Section 4.2.4 mints a halving per-timeslot amount to block proposers, capped in total at 4 Million QBTC, less the longer the bridge stays open (Section 4.2.4), and ending outright once it is delivered. Total issuance is therefore bounded above by $\sim$7.9 Million QBTC. A third quantity exists but is not part of that total: the 50 QBTC created in the genesis block to give the chain its initial validation weight, which consensus permits to be staked but never spent, and which therefore never enters circulation (Section 5.2). The bridge may close before its cap is filled, and legacy coins left unmigrated at that point can never be bridged, so the realized supply may fall well below that bound. There is no discretionary issuance and no perpetual inflation: both components are fixed at the Genesis Upgrade, and both terminate. This monetary policy represents a transition from Phase 1: Distribution (Legacy PoW), where computational work arbitrated initial issuance, to Phase 2: Stewardship (qBitcoin PoS), where the stake established during the distribution phase secures the network without further energy overhead.

To facilitate the PoW-to-PoS transition, qBitcoin implements a pegged bidirectional bridge [7], as illustrated in Figure 2. The upgrade is initiated by locking legacy BTC in reserve addresses managed by the legally accountable Conversion Service, where the funds remain locked unless an inverse downgrade process is initiated. Much as the legacy chain paid its earliest miners the largest block subsidies, the bridge gives its earliest upgraders the best rate: the more legacy BTC has already been upgraded, the fewer QBTC the next holder receives for each BTC locked.

4.1.1 The Conversion Rate Schedule

The conversion rate is set by upgraded volume, not by elapsed time. Write V for the legacy BTC standing in the bridge reserve, which every lock raises and every completed downgrade lowers, and $\Delta = 4{,}200$ BTC for the level size, a protocol constant (Table 2). The bridge's current *level* is

$$L(V) = \lfloor V/\Delta \rfloor \tag{1}$$

Each level carries a conversion rate fixed at the Genesis Upgrade in a table of 5,000 entries, one for every level from 0 to 4,999, stored in fixed point with a precision of 10^{-6}. The rate starts at one QBTC per BTC and falls by 0.1% with every level:

$$m(L) = 10^{-6} \left\lfloor 10^{6} \cdot \delta^{L} \right\rfloor, \qquad \delta = 0.999 \tag{2}$$

where $m(L)$ is the QBTC minted per BTC locked at level L. A lock is converted coin by coin rather than at the rate of the level in which it begins: a lock of X BTC applied at cumulative volume V mints

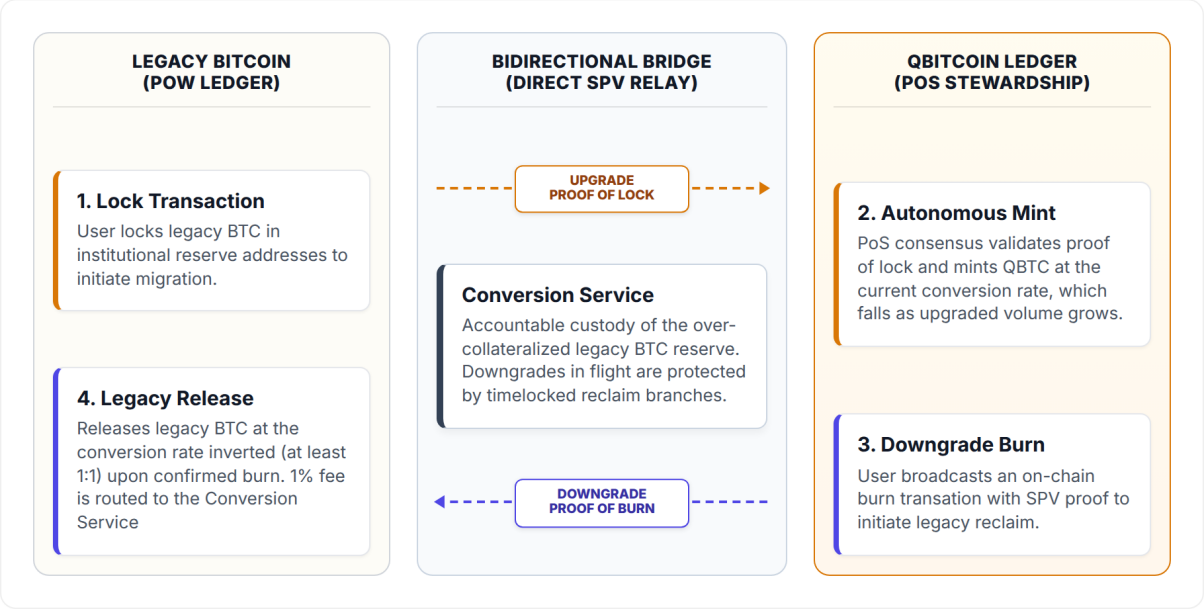


Figure 2: The qBitcoin bidirectional bridge and migration mechanism. Users lock legacy PoW coins in reserve addresses to securely mint post-quantum QBTC on the Proof-of-Stake ledger.

$$Q(V, X) = \int_{V}^{V+X} m\big(L(v)\big)\, dv \tag{3}$$

that is, the rate of every level the lock spans, weighted by the part of the lock that falls inside it. Because the integrand depends only on cumulative volume, the amount minted against a given volume does not depend on how that volume was divided into locks. Splitting one lock into many, or combining many into one, mints exactly the same total, so nothing is gained by rushing a large deposit into a level with a better rate. Of the minted amount, exactly 1% is credited to the Network Reward Fund and the remainder to the holder:

$$Q_{\text{holder}} = 0.99 \cdot Q(V, X), \qquad Q_{\text{fund}} = 0.01 \cdot Q(V, X) \tag{4}$$

Table 3 lists milestones of the schedule and Figure 3 plots it in full. The rate halves at level 693, once 2,910,600 BTC have been upgraded, and the last level that the upgrade cap admits, level 2,499, converts at 0.08 QBTC per BTC.

Table 3: Milestones of the conversion rate schedule ($\Delta = 4,200$ BTC per level, $\delta = 0.999$). $m(L)$ is the QBTC minted per BTC locked; $1/m(L)$ is the BTC that consequently stands behind each QBTC minted at that level.

Level L	**Opens at** V **(BTC)**	$m(L)$	$1/m(L)$	**Note**
0	0	1.000000	1.000	launch
100	420,000	0.904792	1.105	
250	1,050,000	0.778703	1.284	
500	2,100,000	0.606378	1.649	
693	2,910,600	0.499900	2.000	rate halves
2,499	10,495,800	0.082064	12.186	last before cap

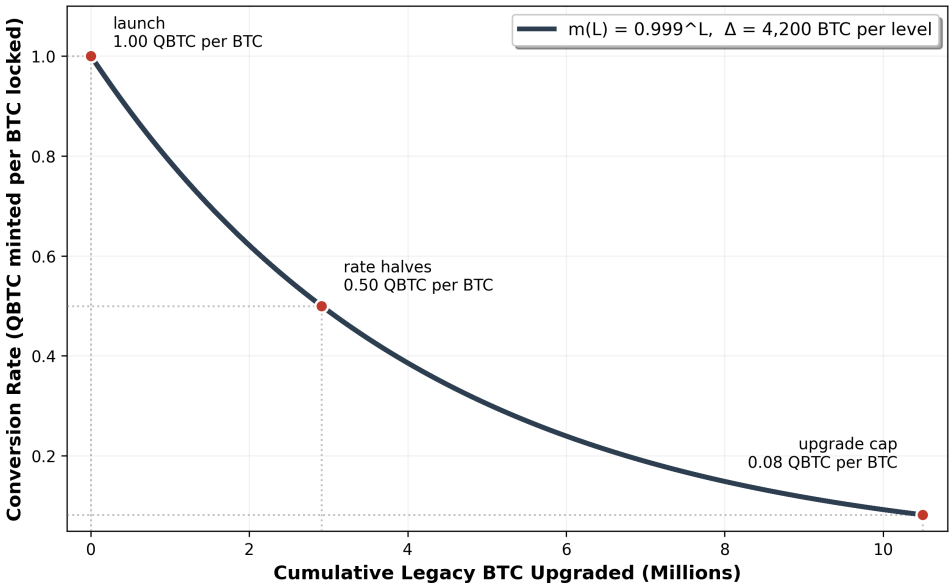


Figure 3: The conversion rate $m(L)$ against cumulative BTC upgraded, from 1 QBTC per BTC at launch to 0.08 QBTC per BTC at the upgrade cap of 10,500,000 BTC ($\Delta = 4{,}200$ BTC, $\delta = 0.999$). The rate halves at level 693.

Setting the rate by volume rather than by time has two consequences. First, a quiet period costs no one anything: the rate offered to the next holder depends only on how much has already crossed the bridge, not on how long ago the Genesis Upgrade took place, so a holder who waits while nobody else upgrades loses nothing by waiting. Second, the bridged supply is fixed by the schedule rather than by the timing of demand. Integrating Eq. 3 over the whole upgrade cap $V_{\max}$ gives the bound quoted above:

$$S_b \;\le\; Q(0, V_{\max}) \;=\; 3{,}855{,}669 \text{ QBTC} \;<\; \frac{\Delta}{1-\delta} \;=\; 4{,}200{,}000 \text{ QBTC} \tag{5}$$

The right-hand bound is the geometric series summed over every level, and it holds whatever the cap: however much BTC the bridge were to accept, it could never mint more than 4,200,000 QBTC, the amount that 1,000 levels would mint at the launch rate.

The falling rate has a consequence for the coins already minted, and that consequence is the bootstrap incentive. A holder who upgrades 1 BTC at launch receives 0.99 QBTC. Once 2,910,600 BTC have crossed the bridge, the rate has halved, and a new holder locking the same 1 BTC receives only about half as much. Seen from the other side, obtaining the early holder's 0.99 QBTC through the bridge now takes 2.00 BTC, the conversion fee included, so the early holding redeems through the bridge itself for about 2.00 BTC, with no buyer to be found for it. The BTC that the lower rate leaves unconverted is neither minted nor paid away: it stays in the reserve, and the downgrade path is what turns it into the early holder's claim (Section 4.2.6). The growth in what an early QBTC is worth in BTC is therefore not a rule of the protocol but a result of one. It follows from the falling rate, and the earlier a holder upgrades, the larger it becomes.

4.1.2 Upgrade Stop Conditions

The migration runs to a bounded end. The bridge closes at the first of three events, and the moment it closes is written T_{close}:

$$T_{\text{close}} = \min\bigl(T_{\text{conv}},\ T_{\text{cap}},\ T_{\text{veto}}\bigr) \tag{6}$$

- **Time, T_{conv}.** The bridge operates for at most 10 years from the Genesis Upgrade, a protocol constant (Table 2) denominated in timeslots rather than blocks, so that it advances on wall-clock time whether or not blocks are being produced.
- **Volume, T_{cap}.** Minting stops in the block in which the reserve volume V reaches the upgrade cap $V_{\max} = 10{,}500{,}000$ BTC, half of the legacy supply. The cap binds the volume standing in the reserve rather than the total ever locked, so it bounds the bridged supply at every moment rather than the number of times a coin may cross: a redemption that takes V back below the cap admits locks again until it refills. A lock that carries V past the cap is minted in full, at the rate of the level it reaches.
- **Veto, T_{veto}.** The bridge closes at the block that confirms an SPV proof, at the confirmation depth required of the downgrade burn (Section 4.3), that any outpoint of the veto set in Table 4 has been spent on the legacy chain. Locking such an output in the bridge is itself a spend, so an attempt to upgrade it triggers the same close.

The veto set consists of six legacy outputs fixed at the Genesis Upgrade. Four of them date from the first weeks of the network in early 2009: the coinbase outputs of blocks 1 to 3 and an 800 BTC output of block 3307. Outputs of that era are conventionally attributed to Bitcoin's pseudonymous creator, and all four pay to bare public keys (P2PK), so their keys have been exposed on-chain since the day they were created. That places them in exactly the class of outputs that a quantum adversary running Shor's algorithm could attack first. A fifth pays to the public key whose 32-byte coordinate is the byte `0x02` repeated: a valid curve point for which no one is expected to hold the private key, so spending it would require solving the discrete logarithm outright. The movement of any of these outputs therefore admits two readings. Either their owner has chosen to act, and the set gives that owner a veto over the migration, or the elliptic-curve security of the legacy chain has failed, and the bridge must stop accepting legacy coins whose ownership can no longer be established by signature. Both readings call for the same response.

Table 4: The veto set: legacy outputs whose spending closes the bridge. Fixed at the Genesis Upgrade.

Outpoint (`txid:vout`)	**Legacy output**
`6c3efe515b5017c5020e1f20a2c5924fa09e6648cf3eb3858771d2cad7edec45:0`	0.1 BTC to public key `02...02` (33 bytes)
`0e3e2357e806b6cdb1f70b54c3a3a17b6714ee1f0e68bebb44a74b1efd512098:0`	50 BTC coinbase (P2PK), block 1
`9b0fc92260312ce44e74ef369f5c66bbb85848f2eddd5a7a1cde251e54ccfdd5:0`	50 BTC coinbase (P2PK), block 2
`999e1c837c76a1b7fbb7e57baf87b309960f5ffefbf2a9b95dd890602272f644:0`	50 BTC coinbase (P2PK), block 3
`44794602bebb5d51996e3c9b6ba9bd72f62bc9308b4eeaf52ca670b0fb9598b4:0`	1 BTC (P2PKH)
`277951bc92bc86ed75ae4baa2ac5e7d1f3ecd2c951819796ad7d04cda431f430:0`	800 BTC (P2PK), block 3307, February 2009

The window and the veto are one-way latches: once either has closed the bridge it stays closed, no further legacy BTC may be locked, and no further downgrade may be initiated. The cap is a condition on the standing volume instead, so a redemption below it reopens minting until it refills. Downgrades already in flight settle under the state machine of Section 4.3, reclaim guardrails included, and once the last of them terminates the reserve is final. The close is also the activation point of the terminal emission schedule (Section 4.2.4): the protocol begins minting exactly when the conversion fees that seeded the fund stop arriving, and where the bridge is held closed by the cap alone, a redemption that reopens minting suspends the emission until the cap is reached again. Holders should read T_{conv} as an upper bound on the window rather than as its length, since the cap may fill, and a veto may occur, at any time.

4.2 The Network Reward Fund: Consolidated Economics & Subsidies

Every bridge mint credits exactly 1% of the minted QBTC to the Network Reward Fund (Eq. 4). For a block h that applies locks of $X_{h,1}, \ldots, X_{h,n}$ BTC at cumulative volumes $V_{h,1}, \ldots, V_{h,n}$, the conversion inflow is

$$C_h = 0.01 \sum_{i=1}^{n} Q\big(V_{h,i}, X_{h,i}\big) \tag{7}$$

The fund releases these coins back into active circulation through the per-block subsidy that constitutes the entirety of validator compensation, as illustrated in Figure 4.

The fee is levied on what is minted, not on what is locked. The part of a deposit that the falling rate leaves unconverted is neither minted nor routed to the fund; it stays in the reserve and backs the coins already in circulation (Section 4.2.6). The conversion fee ensures that every migration directly subsidizes the new network, capitalizing a permanent endowment for validators without perpetual inflation. It is one of three inflows into a single undivided pot, alongside transaction fees and slashing fines. Those three are the whole of the pot's income; the terminal emission that succeeds the fee when the bridge closes reaches validators by a separate route, paid straight to block proposers.

Legacy Bitcoin pays miners from a block subsidy that falls to zero, leaving the security budget to an unsmoothed fee market. qBitcoin separates the two questions. Every unit of value the protocol collects from users — fees, the conversion fee, and slashing fines — is pooled and metered out at a fixed rate, so the size of the security budget and the volatility of what a validator actually earns are set by different mechanisms. The bounded emission is handled differently, for reasons Section 4.2.4 gives: being constant within a halving era, it is already smooth, and it is paid directly. This subsection defines the pot and its payout; Section 4.2.4 defines the emission that keeps it fed after the migration ends.

4.2.1 A Single Pot and a Single Payout

All transaction fees of a block are routed to the Network Reward Fund; no portion is paid directly to the validator. The remaining inflows enter on the same footing: the conversion fee levied on bridge mints (Eq. 7) and the fines levied by slashing transactions (Section 5.6). The terminal emission of Section 4.2.4 is not among them. Writing $B(h)$ for the fund balance after block h, the block's total inflow is

$$I_h = F_h + C_h + \Phi_h \tag{8}$$

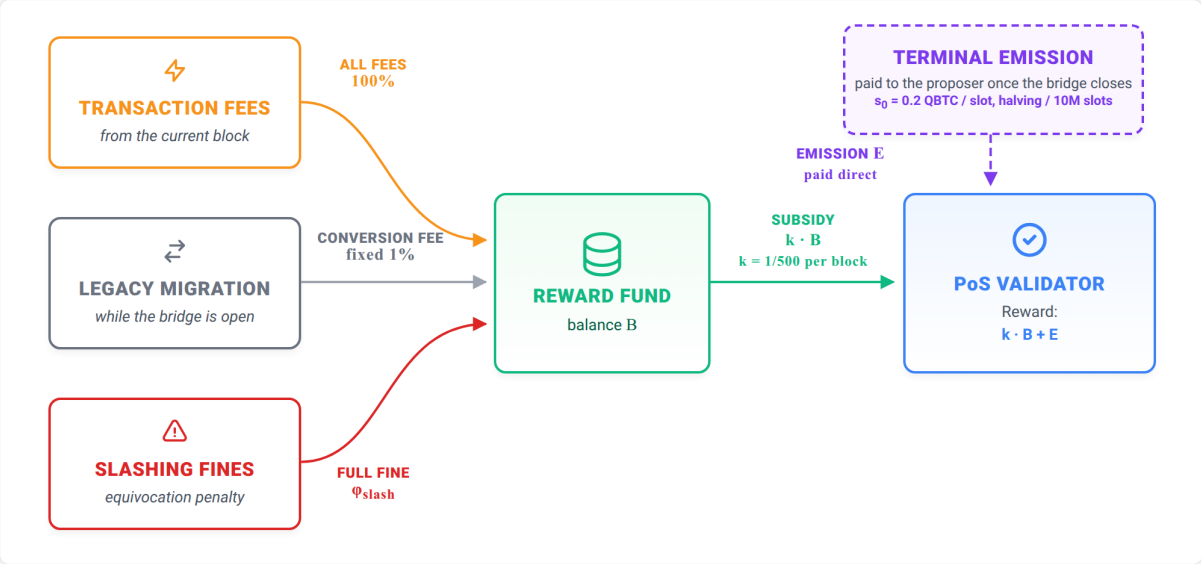


Figure 4: The qBitcoin Reward Fund dynamics. Transaction fees, the 1% conversion fee on bridge mints, and slashing fines are pooled without any split, and the fund pays out a fixed fraction k of its balance per block: an exponential moving average of network income. Once the bridge closes, the terminal emission of Section 4.2.4 is paid directly to each eligible block's proposer, bypassing the pot.

for transaction fees F_h, conversion fees C_h, and slashing fines Φ_h. Inflows are credited before the payout, so a block's own fees are part of the pot it draws from. Write $B^*(h)$ for that pre-payout balance:

$$B^*(h) = B(h-1) + I_h \tag{9}$$

The winning validator is paid a fixed fraction k of $B^*(h)$, together with the block's terminal emission E_h, which the protocol mints directly to the proposer rather than routing it through the fund; the fund retains the remainder of the pot. One class of block is excluded from both, as set out below the equations:

$$R_{\text{validator}}(h) = \begin{cases} 0 & \text{if block } h \text{ is subsidy-ineligible} \\ k \cdot B^*(h) + E_h & \text{otherwise} \end{cases} \tag{10}$$

$$B(h) = \begin{cases} B^*(h) & \text{if block } h \text{ is subsidy-ineligible} \\ B^*(h)\,(1-k) & \text{otherwise} \end{cases} \tag{11}$$

A block is *subsidy-ineligible* in exactly one case, defined in Section 5.4: it is non-mandatory and its entire user payload is the single permitted zero-fee transaction. Such a block carries no fees and, by that definition, no slashing transaction either; the emission is likewise withheld from it (Eq. 14), so $I_h = 0$ and the fund is left untouched rather than following a separate accounting path.

Two details of the pooled payout are exact in the protocol and immaterial at any realistic balance. It is truncated to the smallest indivisible unit, and it is floored at one satoshi whenever the fund is non-empty, so a fund left without inflow drains linearly at its tail rather than approaching dust asymptotically.

The subsidy fraction k is fixed at the Genesis Upgrade and set to $1/500$ per block (Table 2). The fund starts empty, $B(0) = 0$; early validator rewards therefore track the arrival of conversion fees until a recurring fee base is established, and the emission does not begin until the bridge closes.

4.2.2 The Fund as an Exponential Moving Average

Unrolling Eq. 11 over subsidy-eligible blocks gives the pooled part of the payout, $R_{\text{pool}} = kB^*$, as an exponential moving average of protocol income with smoothing factor k; an ineligible block has $I_h = 0$ and leaves B unchanged, so it is a no-op in the recursion and does not disturb the identity:

$$R_{\text{pool}}(h) = k \sum_{j \geq 0} (1-k)^j \, I_{h-j}, \qquad k \sum_{j \geq 0} (1-k)^j = 1 \tag{12}$$

The weights sum to unity: the fund delays value without creating or destroying it. Every closed form in this subsection describes the pooled component of validator income alone; the terminal emission is not an inflow, does not enter the recursion, and is added to the pooled payout as a term that is constant within a halving era (Eq. 10). Deviations from the steady state B_∞ decay geometrically, with no approximation, since the substitution $(B_\infty + I)(1-k) = B_\infty$ cancels identically in Eq. 11:

$$B(h) - B_\infty = \big(B(h_0) - B_\infty\big)(1-k)^{\,h-h_0} \tag{13}$$

Setting $\tau = 1/k = 500$ blocks gives $(1-k)^\tau = 0.3675$, within 0.1% of e^{-1}. Thus τ is the fund's time constant in the standard sense: any disturbance — a fee spike, a large migration settlement, a collapse in demand — is absorbed at a rate of one factor of e per 500 blocks, and 63.2% of the gap to the new steady state is closed within τ. Table 5 collects the closed-form properties.

Table 5: Closed-form properties of the Reward Fund recursion.

Property	**Expression**	**Value at** $k = 1/500$
Steady-state balance	$I\,(1-k)/k$	$499\,I$
Steady-state pooled payout	I	payout equals inflow
Time constant	$\tau = 1/k$	500 blocks
Residual deviation after τ	$(1-k)^{1/k}$	$0.368\ (= e^{-1})$
Residual after 2τ / 3τ	$(1-k)^{n/k}$	0.135 / 0.050
Half-life	$\tau \ln 2$	346 blocks
Weight centre of mass	$(1-k)/k$	499 blocks
Pooled payout standard deviation	$\sigma\sqrt{k/(2-k)}$	$\sigma/31.6$

The horizon this implies should be read literally. Under full-cadence production, $\tau = 500$ blocks is 83 minutes and the half-life is 58 minutes, so the fund smooths intra-hour lumpiness in the fee stream, not daily or weekly fee cycles: a fee spike is forgotten within a few hours, not a few weeks. That is the intended trade. A longer τ would smooth deeper cycles, but it would also park proportionally more supply in the fund and make the low-demand behaviour of Section 4.2.3 correspondingly slower to respond.

4.2.3 Activity-Scaled Distribution

The parameter k is denominated in blocks, and block production is demand-driven: under load every 10-second timeslot is filled, while under no load only the mandatory heartbeats are produced, one per 100 timeslots (Section 5.4). The time constant is therefore fixed in block count and variable in wall-clock time.

Table 6: Wall-clock length of the time constant under the two block-production regimes.

Regime	**Cadence**	$\tau = 500$ **blocks**	**Pooled payout after 1 day of zero fee inflow**
Under load	10 s	83 min	3×10^{-8}
Heartbeat only	16.7 min	5.8 days	0.84

The ratio is the heartbeat divisor, 100, and it requires no retargeting mechanism, external oracle, or governance action: the balance accumulated under load is disbursed over a proportionally longer interval, because the drain is per block and blocks have become rare. Heartbeats are then the sole claimants. With zero inflow the per-block reward retains 84%, 30%, and 0.6% of its pre-drought level after a day, a week, and a month of heartbeats.

Table 6 describes decay toward zero, and the decay is real: the emission is paid to proposers rather than into the fund, so a sufficiently long fee drought drains the pool toward nothing. The emission runs through the drought regardless. Once it activates (Section 4.2.4) every subsidy-eligible block carries E_h whatever the fund holds, so the residual figures above describe the decay of a validator's *pooled* income, beneath which the emission floor remains.

The three pooled inflows differ in duration, and the emission that succeeds them is of a fourth kind again. The conversion fee is levied once per migrated balance, so its contribution is a transient: at $\tau = 500$ blocks the fund retains no memory of it beyond a few hours of full-cadence production, and the flow stops when the bridge closes. Slashing fines are by design rare. The conversion fee should therefore be read as bootstrap capital for the transition rather than as a durable pillar of the security budget. That is precisely why the terminal emission of Section 4.2.4 begins where it ends. The emission is itself finite, halving out over roughly 79 years, so in the long run the fund is fed by transaction fees alone; what the schedule buys is that the handover to a mature fee market is gradual rather than a cliff.

4.2.4 The Terminal Emission Schedule

The conversion fee is bootstrap capital with a bounded end date: when the bridge closes at T_{close} (Section 4.1.2), that inflow stops, and a network whose fee base is still maturing would be left to an unsmoothed fee market at exactly the moment its endowment stops growing. The terminal emission schedule spans that gap. Beginning at the close of the bridge, the protocol mints at a fixed rate per timeslot and pays each eligible block the amount accrued since its predecessor; the rate halves at a fixed interval of timeslots, so the schedule delivers a bounded total and then stops permanently.

Write u_h for the timeslot index of block h, counted from the Genesis Upgrade, u_{start} for the timeslot at which the bridge closes, $H_{\text{halve}} = 10{,}000{,}000$ timeslots for the halving interval, and $s_0 = 0.2$ QBTC for the initial emission per timeslot. The interval and the rate are protocol constants fixed at the Genesis Upgrade (Table 2). A block is paid for the slots elapsed since its predecessor, $\Delta_h = u_h - u_{h-1}$, so the emission it carries is

$$E_h = \begin{cases} 0 & \text{if } u_h < u_{\text{start}} \\ 0 & \text{if block } h \text{ is subsidy-ineligible} \\ \lfloor s_0 \cdot 2^{-n_h} \rfloor \cdot \Delta_h & \text{otherwise} \end{cases} \tag{14}$$

where $n_h = \lfloor u_h / H_{\text{halve}} \rfloor$ is the halving era block h falls in and the floor is taken in the smallest indivisible unit, so the schedule terminates outright: after 25 halvings the per-timeslot amount truncates to zero and no further coin is ever created. The era index runs from the Genesis Upgrade rather than from the close, so the schedule is a fixed calendar which the closing of the bridge switches on partway through: whatever it would have paid before u_{start} is never paid at all. Summing the geometric series over the slots of each era bounds the total:

$$\sum_{u_h \geq u_{\text{start}}} E_h \;\leq\; s_0 \, H_{\text{halve}} \sum_{n \geq 0} 2^{-n} \;=\; 2 \, s_0 \, H_{\text{halve}} \;=\; 4 \text{ Million QBTC} \tag{15}$$

The bound is reached only by a bridge that closes at the Genesis Upgrade itself; every year the window stays open spends part of the calendar unpaid, and Table 8 states what each closing point leaves of it. Two further deductions are minor beside that: integer truncation costs a few coins across the whole schedule, and the slots covered by a subsidy-ineligible block go unpaid. Table 7 states the schedule's own figures.

Table 7: The terminal emission schedule.

Property	**Expression**	**Value**
Activation	T_{close}	bridge close ($\leq$ 10 years)
Initial emission	s_0	0.2 QBTC per timeslot
Halving interval	H_{halve}	10,000,000 timeslots
Era length	$H_{\text{halve}} \cdot 10\,\text{s}$	$\approx$3.2 years
Emitting eras	until $\lfloor s_0 2^{-n} \rfloor = 0$	25
Schedule length (from genesis)	$25\, H_{\text{halve}}$	250,000,000 slots ($\approx$79 years)
Total emitted	$2 s_0 H_{\text{halve}}$	$\leq$ 4 Million QBTC (Table 8)
First-year emission (era 0)	$s_0 \cdot 3{,}155{,}760$	631,152 QBTC ($\approx$16.4% of the bridged cap)
Terminal supply bound	bridged + emitted	$\sim$7.9 Million QBTC

Table 8: What the terminal emission delivers, by the point at which the bridge closes. The halving era is counted from the Genesis Upgrade, so a longer conversion window leaves less of the schedule to pay out.

Bridge closes (years after genesis)	**Era** n	s_n **(QBTC/slot)**	**Emitted thereafter (QBTC)**
0	0	0.2	3,999,999
1	0	0.2	3,368,847
3	0	0.2	2,106,543
5	1	0.1	1,422,119
8	2	0.05	737,695
10	3	0.025	461,059

Two properties of Eq. 14 are deliberate and worth stating explicitly.

The schedule runs on wall-clock time. Both halves of Eq. 14 are counted in timeslots: the era index n_h measures slots elapsed since the Genesis Upgrade, and each block is paid for the slots it covers. The issuance rate is therefore s_n per ten seconds at any cadence, and a block closing a gap mints for the gap: a heartbeat block spanning 100 slots carries $100\, s_n$, a full-cadence block carries s_n. Every wall-clock figure in Table 7 is a property of the schedule rather than a projection, an era spanning 3.2 years however busy the network is; it is the block counts that vary with cadence, since the same 250,000,000 slots are covered by as many blocks under load and by a hundredth as many in a network producing only heartbeats. The denomination differs from the fund's, where k is applied per block (Section 4.2.3): a quiet period stretches the pooled payout in wall-clock terms while leaving the emission on the clock, so a validator producing heartbeats through it is paid emission for the whole interval each block covers.

Ineligible blocks emit nothing. A subsidy-ineligible block has $I_h = 0$ and leaves the fund untouched, a no-op in the recursion of Eq. 11; withholding the emission from it as well keeps the zero-fee allowance of Section 5.4 from being farmed for freshly minted coin. The slots such a block covers are paid to nobody, which is the shortfall already noted in Eq. 15.

The floor this establishes is the point of the mechanism, and because the emission is paid directly it needs no derivation. Within era n the rate is constant at s_n per slot, so in a network with no fee base at all (one where the pooled term $kB^*(h)$ has decayed to nothing), the proposer of every subsidy-eligible block still receives

$$R_{\text{validator}}(h) \;\geq\; s_n\, \Delta_h, \qquad s_n = \left\lfloor s_0 \cdot 2^{-n} \right\rfloor \qquad \text{for } h \text{ within era } n \tag{16}$$

The bound holds exactly from the first emitting block, since the emission is paid directly rather than accumulated through the fund. In the first era it is 0.2 QBTC per slot, 1,728 QBTC a day across the network whatever the cadence, and it halves each era thereafter. The fund smooths income and the emission creates it, which is why the floor survives a drought that decays the pooled payout to zero, and why it stands before a fee market exists. Two qualifications belong with that. The floor is *predictable*: whether it covers the cost of running a validator depends on the price of QBTC, which the market sets and this paper leaves open. And it is temporary: it halves roughly every 3.2 years, 99.2% of the total is delivered within seven halvings, after which the security budget is fee-funded exactly as it was before. The schedule buys decades of graduated handover.

Finally, the emission is paid straight to the proposer: the validator of an emitting block receives the whole of E_h under Eq. 10, where the same sum arriving as fee income would have yielded it kE_h. The rate is a protocol constant within an era, identical for every eligible block, so the payment passes through unsmoothed. An emitting block is therefore worth s_n per slot it covers to propose: the same premium for every proposer, and independent of what the block carries, so it raises the level of validator income while leaving transaction selection alone.

4.2.5 Soft Fee Priority

Because a block's own fees are credited before its payout is computed (Eq. 9), the proposer's reward does depend on what the block carries — but only weakly. A transaction paying fee f raises the payout of the block that includes it by kf; the remaining $(1-k)f$ is retained by the fund and dispersed across all subsequent winners. The proposer thus internalizes $1/500$ of each fee it collects, against the whole of it under a direct fee-market split.

Both consequences are intended. First, the ordering is preserved: kf is strictly increasing in f, so raising a fee still raises the probability that a rational proposer includes the transaction early, and fee-based priority remains available to senders who need it. Second, the pressure

is attenuated by the factor k — a margin $500\times$ weaker. At that margin, fee differences act as a soft ordering signal rather than the dominant term in block construction: a proposer gains little by displacing low-value transfers to make room for higher bidders, and nothing at all by leaving space empty in the hope of a richer mempool. The outcome is fee priority without a hard fee auction. Low-value and zero-fee transfers retain access to block space, subject to the per-block allowance and the dynamic fee floor of Section 5.4 — a floor that rises as blocks fill, and so bites hardest under precisely the contention this paragraph describes.

The size of a fee confers no consensus weight. Every transaction a block carries adds one unit to the block's weight regardless of type, but that term counts transactions rather than pricing them, and it is negligible beside age-weighted stake. Weight that responds to what a transaction *is* remains reserved for those whose prompt inclusion serves the network rather than the payer: slashing evidence and bridge transactions, which enter the block-weight rule of Eq. 20 as the additive terms W_{slash} and W_{bridge} (Sections 5.6 and 5.4). Raising a fee buys neither.

4.2.6 Supply Accounting and the Reserve Curve

The protocol tracks two disjoint supply components. The *bridged supply* S_b rises with every bridge mint and falls with every completed downgrade burn; the *emitted supply* S_e rises only with Eq. 14. Total supply is $S_b + S_e$, and the two never mix: no emitted coin was ever minted against a locked BTC, and none carries a claim on the reserve. The 50 QBTC genesis stake of Section 5.2 belongs to neither component and is excluded from both: it is stake-only by consensus rule, so it can never be spent into circulation nor presented at the bridge, and the accounting below is unaffected by it.

Bridged supply is not an independent quantity. Both directions of the bridge read the same schedule at the same point, so the reserve and the coins outstanding against it are two views of the single state variable V of Section 4.1.1:

$$\text{Reserve}_{\text{BTC}} = V, \qquad S_b = Q(0, V) = \int_0^V m\big(L(v)\big)\, dv \tag{17}$$

A lock moves V up the schedule and mints the QBTC found along the way; a burn moves it back down and releases the BTC found along the way. Redemption is therefore the forward conversion read backwards: burning q QBTC releases the quantity X of BTC for which $Q(V - X, X) = q$, which for an amount small beside the level size is $q/m\big(L(V)\big)$ BTC. Eq. 17 is preserved by construction rather than maintained by policy.

Three properties follow, and the rest of the bridge rests on them:

1. **Every bridged QBTC is backed by at least one BTC.** A lock adds its full X BTC to the reserve and at most X QBTC to S_b, strictly less at every level above zero, because $m(L) \leq 1$ everywhere. The Reward Fund's 1% is minted within that bound rather than beside it.

2. **The reserve redeems exactly.** Because Eq. 17 is an identity, redeeming the whole bridged supply returns exactly the reserve and leaves $V = 0$. Neither the order of redemptions nor their size can leave the bridge unable to pay a claim it accepts.

3. **A redemption is priced where it leaves the schedule.** The rate is the one standing at the moment of the burn, so the first holder to redeem from a filled cap is paid about 12.19 BTC per QBTC, and every redemption lowers V and with it the rate paid to the next, back toward 1:1 as the reserve empties. The average backing, $V/S_b \approx 2.72$ at a filled

cap, is what the reserve pays across the whole queue rather than what any one holder receives.

The third property is what makes the bridge a two-way market maker rather than a mutual fund, and it should be read plainly: a holder's BTC per QBTC depends on where the volume stands when they burn, so a queue of redemptions works its way down the schedule. The BTC an early holder can realize is largest while the bridge is full and falls as others redeem ahead of them. A holder who locked late and redeems at once recovers their own deposit less the two fees, having entered and left at the same point of the schedule.

Because both sides are quoted at the same point, the bridge also pins what QBTC is worth in BTC for as long as it is open. No buyer need pay more than the BTC the bridge would take to mint the same coins, and no holder need accept less than the bridge would release for them, so for quantities small beside the level size, and up to the two 1% fees,

$$\text{Price}_{\text{QBTC/BTC}} \;\approx\; \frac{1}{m\big(L(V)\big)} \tag{18}$$

a quantity that rises as the schedule advances. That is what makes early conversion pay, and it pays through the bridge itself rather than through a buyer: a holder who upgraded 1 BTC at launch holds 0.99 QBTC, and redeeming it while the schedule stands at the upgrade cap releases about 11.94 BTC, both fees paid. The earliest holders thus acquire the largest claim on the BTC that later holders commit, without any coin being minted to pay for it.

One boundary of this guarantee belongs with it. The consensus rules bound the payment rather than compute it: a commitment is valid only if it pays at least one BTC per QBTC less the service fee (Section 4.3), while the ledger books every redemption at the schedule rate of Eq. 17. The Conversion Service constructs the payment between those two bounds, and a payment departing from the schedule rate leaves the recorded volume and the reserve standing behind it out of step.

The corollary must be read plainly: redeemability is a property of the conversion window, not a permanent feature of QBTC. When the bridge closes, the downgrade path closes with it, the reserve is settled against the downgrades still in flight and then final, and the coins minted thereafter under Eq. 14 carry no claim on legacy BTC. From that point, neither does any other QBTC. A holder who values the option to return to the legacy chain must exercise it before T_{close}, which may arrive well within 10 years of the Genesis Upgrade. Past that point qBitcoin stands on its own ledger, and the $\sim$7.9 Million bound of Section 4.1, not the reserve, is what constrains its supply.

4.3 The Trust-Minimized Downgrade Protocol & Reclaim Guardrails

The qBitcoin protocol supports a downgrade operation that converts QBTC back into legacy BTC. This two-way conversion is facilitated by the Conversion Service: a legally registered, accountable organization responsible for managing the Bitcoin reserve and executing legacy transactions.

4.3.1 Architectural Necessity of the Conversion Service

The bridge is inherently asymmetric: qBitcoin can verify Bitcoin events on its own through Simplified Payment Verification (SPV), but the legacy Bitcoin network cannot verify qBitcoin events in return. Releasing BTC from the reserve therefore cannot be triggered automatically by the qBitcoin ledger and must instead be handled by an accountable operator.

Consequently, the presence of a legally accountable Conversion Service is an operational necessity to manage the physical custody and release of legacy BTC reserves. To sustain this bridging infrastructure, backward conversion incurs a fixed 1% operational fee paid to the Conversion Service. The redemption rule follows the schedule: QBTC burned through the downgrade protocol releases the BTC that the conversion rate of Section 4.1.1 stands at when the burn confirms, read backwards as in Eq. 17, less the 1% service fee, to the holder's specified legacy Bitcoin address. Consensus does not recompute that figure; it requires of every commitment only that it pay at least one BTC per QBTC less the fee, so the schedule rate is the floor's upper companion rather than a second check. The rule holds in aggregate as well as per transaction, because the reserve and the bridged supply move together along one schedule (Section 4.2.6), and because the downgrade path is available only while the bridge is open: a downgrade may be initiated at any point before T_{close} and not after.

4.3.2 The Downgrade State Machine

The downgrade workflow is executed as a multi-step on-chain state machine, ensuring that a temporary outage or technical issue on the Conversion Service's side cannot freeze locked funds:

1. **Freeze Initiation:** To initiate a downgrade, the holder broadcasts a qBitcoin transaction sending QBTC to a protocol-defined Pay-to-Script-Hash (P2SH) Freeze Script. The output data commits to two parameters: a reclaim identifier derived from the holder's public key, and the destination Bitcoin `scriptPubKey`.

2. **Service Commitment:** The Conversion Service spends the holder's freeze output into a specialized `TX_TYPE_DOWNGRADE` transaction, signed by two of the three Conversion Service federation keys. This on-chain commitment creates a new downgrade output with an extended timeout and records the exact legacy Bitcoin payment parameters: the Bitcoin transaction ID, output index, amount, destination `scriptPubKey`, and the original qBitcoin freeze outpoint. The amount is fixed at this step, at or above the one-BTC-per-QBTC floor that consensus enforces and ordinarily at the schedule rate of Eq. 17, so that it is known before any Bitcoin is released and can be checked exactly in the final step.

3. **Bitcoin Release:** Once the `TX_TYPE_DOWNGRADE` commitment is confirmed on qBitcoin, the Conversion Service broadcasts the corresponding release transaction on the legacy Bitcoin network. This transaction pays the committed amount to the destination address and includes an `OP_RETURN` marker containing the original qBitcoin freeze outpoint, preventing replay attacks or the reuse of historical payments.

4. **SPV Proof & Burn:** After the legacy Bitcoin transaction achieves sufficient confirmation depth, any network participant may broadcast a `TX_TYPE_BURN` transaction on qBitcoin. This transaction spends the downgrade output through a permissionless branch, carrying an SPV proof of the Bitcoin block inclusion. Consensus validation checks that the transaction ID matches the commitment, the payment satisfies the committed amount and destination, and the outpoint marker is present. Upon successful verification, the locked QBTC is permanently burned, finalizing the downgrade.

4.3.3 The Reclaim Path: Fallback Guardrails

So that an outage or technical issue on the Conversion Service's side does not leave the funds frozen, the Freeze Script and the downgrade output that succeeds it are P2SH redeem scripts of

one shared shape: a service branch guarded by OP_IF, and a holder branch behind OP_ELSE that a relative timelock opens. Both branches are written out in the script itself rather than committed as a Merkelized Abstract Syntax Tree (Section 6.3). With only two of them, and both fixed by the protocol for every downgrade in the network, a Merkle commitment would conceal nothing that is not already public and would cost an inclusion proof to spend. The shared shape is

```
OP_IF <service branch> OP_ELSE <timeout> OP_CSV OP_DROP
        <reclaim identity check> OP_ENDIF
```

and the two branches carry the following conditions:

- **Branch A (Service Execution):** On the freeze output, spendable only in a TX_TYPE_DOWNGRADE transaction, and then only by two of the three Conversion Service federation keys. Pinning a holder's freeze to a payment that never follows therefore requires compromising two operators rather than one. On the downgrade output that the commitment creates, the same branch is permissionless and carries no signature at all: it opens to any TX_TYPE_BURN, whose correctness rests on the SPV proof rather than on who broadcasts it.
- **Branch B (Holder Reclaim):** Governed by a relative timelock (OP_CHECKSEQUENCEVERIFY) bound to the holder's reclaim identifier. The lock counts wall-clock time rather than blocks, since qBitcoin skips empty timeslots. The script reads the reclaim identifier out of its own output data and requires a signature from the public key that hashes to it, so the branch is open to the holder alone and needs no cooperation from anyone.

If the Conversion Service experiences an outage or technical fault and does not broadcast a valid commitment within 48 hours, Branch B becomes active. The holder broadcasts a reclaim transaction, canceling the downgrade and recovering 100% of their locked QBTC.

Similarly, if the Service creates a TX_TYPE_DOWNGRADE commitment but fails to broadcast a valid, SPV-verifiable Bitcoin payment within seven days, the downgrade output defaults back to the holder's reclaim path. Thus, the protocol guarantees two terminal outcomes: either a valid Bitcoin payment is cryptographically proven and the QBTC is burned, or the holder retains full ownership of their funds.

5 Network Stewardship and Consensus: UTXO-based Proof-of-Stake

With the network capitalized and validators incentivized through the economic mechanisms outlined in Section 4, qBitcoin relies on a UTXO-based Proof-of-Stake (PoS) mechanism to secure the ledger. Under the PoS design, stake weight is calculated as value $\times$ age (in blocks) for each staking UTXO, as formalized in Eq. 19, decoupling network security from energy-intensive computational grinding.

5.1 Block Generation and Stake Weight

Block production proceeds in discrete *timeslots*: time is partitioned into consecutive 10-second intervals matching the target block interval (Table 2), and at most one block may be appended to the chain per timeslot.

In the qBitcoin UTXO-based Proof-of-Stake model, validators do not compete by expending computational energy to find a target hash. Instead, a validator proposes a block for the current timeslot by constructing a specific "stake transaction" (T_{stake}), which is always the first transaction of the block and authenticates it. The stake signature is computed over a message binding three elements: (i) the hash of the previous block, (ii) the block's timeslot, and (iii) the hash of the block's remaining transaction set. Because the timeslot is part of the signed message, a stake signature is bound to a specific (staked output, timeslot) pair; an honest validator signs at most one block per timeslot for any given staked output - the property underpinning the slashing mechanism of Section 5.6. The protocol deterministically evaluates the block's stake weight based strictly on the UTXOs consumed in that transaction.

The stake weight (W_{stake}) of a proposed block is defined as the sum of value $\times$ age products for each input UTXO used in the stake transaction, formalized in Eq. 19:

$$W_{\text{stake}} = \sum_{i=1}^{n} (v_i \cdot a_i) \tag{19}$$

Where:

- W_{stake} is the total calculated stake weight for the proposed block.
- n is the total number of UTXOs provided as inputs to T_{stake}.
- v_i is the exact nominal value of the i-th UTXO (measured in QBTC).
- a_i is the age of the i-th UTXO, defined as the difference between the current block height (H_{current}) and the block height at which the UTXO was minted ($H_{\text{mint_i}}$).

Stake weight is one of four terms in a block's weight. The effective weight used to arbitrate a timeslot adds three further contributions, all defined below (the bridge term W_{bridge} of Section 5.4, the slashing term W_{slash} of Section 5.6, and a count N_h of the transactions the block carries), and is given in full by Eq. 20:

$$\begin{aligned} W_{\text{block}}(h) &= W_{\text{stake}}(h) + W_{\text{slash}}(h) + W_{\text{bridge}}(h) + N_h \\ W_{\text{bridge}}(h) &= \sum_{j \in \mathcal{B}(h)} v_j \cdot A \cdot \rho_j, \qquad \rho_j = \begin{cases} \dfrac{A}{A + d_j} & \text{for an upgrade} \\ 1 & \text{for a redemption} \end{cases} \end{aligned} \tag{20}$$

By defining the canonical block for a given timeslot as the candidate with the highest effective weight W_{block} (Eq. 20), with exact ties broken by the lower block hash so that the choice is deterministic for every node holding the same candidate set — and by accumulating that same quantity into chain weight (Section 5.5), so that slot arbitration and fork choice always agree — the protocol eliminates computational grinding. Because v_i and a_i are fixed historical facts recorded on the ledger, validators cannot alter arbitrary data (such as nonces or timestamps) to generate a more favorable hash. Block weight does remain sensitive to transaction selection, through the slashing and bridge terms of Eq. 20; that sensitivity is the deliberate inclusion incentive for those transaction types, and it is bounded by what the mempool actually contains. Furthermore, upon successfully proposing a block, the staking UTXOs are consumed; the returned funds and newly minted rewards are issued as new UTXOs with an age of zero, subject to a maturity lock that prevents them from being spent for a protocol-defined interval (Table 2). The lock governs ordinary spends only: a fresh stake output may be re-staked at once, and

what limits re-staking is the age reset rather than the lock, since a zero-age output contributes no weight until it has aged. The coin-age depletion naturally cycles network power, ensuring no single entity can permanently dominate block production without continuously locking liquidity.

5.2 The Genesis Stake

A proof-of-stake chain cannot validate its first block with stake that does not yet exist. At the Genesis Upgrade the ledger holds no upgraded coins at all, because the bridge has not yet run, so every candidate block would carry zero weight under Eq. 20 and none could be canonical. qBitcoin resolves this the way the legacy chain resolved its own first block, with a fixed issuance written into the genesis block itself: an inputless stake transaction creates 50 QBTC and nothing else, and the chain has a non-zero validation weight from that block onward.

These are service coins, and the protocol treats them as a permanent fixture of the ledger rather than as money. No legacy BTC backs them, since none had been locked before the chain existed, so letting them circulate would break the guarantee that every circulating QBTC can be redeemed from the reserve. A consensus rule keeps the two apart directly: no transaction may decrease the balance held by a genesis output's script, so whatever value a transaction takes from such a script must return to the same script in its outputs. The genesis coins can therefore be staked, which returns their full value, and split among several validator nodes, which is how the bootstrap weight is distributed. They cannot be spent, redeemed through the bridge, or slashed. Here qBitcoin and the legacy chain differ in kind: the 50 BTC of the legacy genesis block is unspendable through an accident of that implementation, and sits inert, whereas the 50 QBTC is unspendable by rule and works for its keep.

Three consequences deserve statement. The genesis stake sits outside the supply accounting of Section 4.2.6: the protocol does not count it as minted and it never enters circulation, so the supply bounds of that section hold as stated and are not raised by it. It is exempt from the slashing of Section 5.6, because a fine cannot be taken from a balance that may not decrease. And it is the only stake in existence until the first upgraded coins have aged, so it dominates block weight over the chain's opening interval, after which ordinary staked coins overtake it and 50 QBTC becomes negligible beside a bridged supply in the millions.

5.3 Delegated Staking

An owner may delegate staking authority over a set of outputs to another node without surrendering the ability to spend them: the node that signs need not be the party that owns the coins.

Delegation is a covenant over the coins. The delegated outputs are bound by a script that lets the delegate's key use them in a stake transaction alone, so the delegate proposes blocks with the coins while spending authority stays with the owner, who withdraws them unilaterally. Delegation keys carry dedicated version bytes, which keeps a key authorized to stake distinct from one authorized to spend. The delegate keeps a configured share of the block reward, paid to an address it nominates, and the remainder accrues to the owner.

Two consequences follow. Stake pools form by this route, so block production may be more concentrated than coin ownership, and the coin-age rotation of Section 5.1 acts on the coins rather than on the operators staking them. And because the delegate signs, a faulty or dishonest delegate can equivocate with coins it does not own, leaving the fine of Section 5.6 to

fall on the owner; the protocol makes that failure fail-stop rather than repeatable, so the loss is bounded rather than compounding.

5.4 Network Health and Transaction Prioritization

Because validators are compensated by the pooled per-block subsidy of Section 4.2, and not by a direct fee share, the protocol supports network health features that would be difficult to sustain in a pure, un-smoothed fee-market system:

- **Mandatory Blocks (Heartbeat Protocol):** To prevent the chain from stalling during periods of exceptionally low activity, a block is mandatorily generated every 100th timeslot (approximately 16 minutes). Mandatory blocks may be empty (containing no transactions besides their stake transaction) and remain reward-eligible under Reward Fund subsidy rules: such a block collects no fees of its own, but still draws the full fraction k of the fund balance. The heartbeat cadence is also what stretches the fund's distribution horizon during quiet periods, as described in Section 4.2.3. Empty blocks produced outside mandatory timeslots are considered invalid and are rejected by the network.

- **Low-Fee Transaction Allowance:** The protocol rules permit the inclusion of exactly one transaction per block that pays either no fee at all or a fee below the prevailing minimum rate; the two share a single slot, so one below-floor payer consumes the allowance as surely as a zero-fee transfer does. The cap counts user transactions only: the block's stake transaction and any slashing transactions it carries are protocol transactions, bear no fee by construction, and fall outside the cap. A bridge transaction is not zero-fee for this purpose — the conversion fee it pays counts as a fee, even though Eq. 8 books it as C_h rather than F_h — so the allowance never limits how many such transactions a block may carry. To prevent this allowance from being exploited to farm the subsidy, a non-mandatory block whose entire user payload is a single zero-fee transaction pays no reward — the subsidy-ineligible case of Eq. 10 — and its stake transaction must not claim the subsidy. A block that also carries a slashing transaction stays eligible, since it delivers protocol value and an inflow Φ_h. Because proposing an ineligible block still consumes the coin-age of its staked inputs, producing it is economically irrational, confining the allowance to its intended purpose of subsidizing genuine low-value transfers within otherwise useful blocks. Ineligibility keys on a genuinely zero fee rather than on the wider low-fee count.

- **Dynamic Minimum Fee:** The floor against which that allowance is measured moves with demand. It rises toward a ceiling as blocks fill and decays by a tenth each block as they empty, from a base rate low enough to be irrelevant in ordinary conditions. It responds to occupancy rather than to what senders bid, so it is a spam control rather than a fee auction, but it does rise when block space is contended. A separate mempool-level cap bounds how many zero-fee transactions a node will hold at once.

- **Bridge Transaction Priority:** Transactions interacting with the bidirectional bridge carry weight of their own, independent of the proposer's stake. This applies both to the locking of legacy PoW coins for forward migration and to the proof of burn that completes the downgrade path. Each enters the block-weight rule of Eq. 20 with the weight its value would carry as stake aged A, one year of timeslots (Table 2), raising the block's chance of winning its timeslot and, through Eq. 21, its branch's cumulative weight. The

term is additive and proportional to value rather than a flat multiplier: a block carrying ten bridge transactions carries the weight of all ten, and a larger conversion counts for more than a smaller one. An upgrade's share is discounted by ρ_j, which decays with the delay between its legacy confirmation and its inclusion, so the incentive is to carry such a transaction promptly rather than to hold it back. The veto marker of Section 4.1.2 is weighted as an upgrade of the entire remaining quota, so that a branch which suppresses it can never outweigh one that carries it. This ensures smooth network transitions and reliable exit paths for capital stewards.

5.5 Network Security: Fork Choice and the Reorganization Penalty

Nodes adopt the valid chain with maximal cumulative weight W_{chain}, the sum of the block weight of Eq. 20 over every block in the branch, as formalized in Eq. 21:

$$W_{\text{chain}}(C) = \sum_{h \in C} W_{\text{block}}(h) = \sum_{h \in C} \big(W_{\text{stake}}(h) + W_{\text{slash}}(h) + W_{\text{bridge}}(h) + N_h \big) \tag{21}$$

Eq. 20 is therefore the protocol's single weight function: the same quantity that arbitrates a timeslot also accumulates into chain weight. This matters because the two rules would otherwise disagree. A block winning its timeslot on its bridge weight but contributing none of it to its branch could be displaced, at no cost, by the very sibling it beat, and since the penalty $P(d_{\text{eff}})$ below engages only once a reorganization reaches further back than the propagation buffer, that displacement would be free for the block's first hundred-odd confirmations. Blocks carrying bridge transactions would then be the most reorganization-prone on the chain, inverting the priority the term exists to provide.

Admitting W_{bridge} into cumulative weight carries a cost, stated here rather than left implicit. The two non-stake terms are of different kinds: W_{slash} is *conserved*, transferring the equivocator's own weight between branches rather than creating any, whereas W_{bridge} is *created* by the protocol as an inclusion incentive, in proportion to the value crossing the bridge. Created weight is in principle a lever on fork choice, and three properties of the bridge blunt it. Bridge transactions are public and deterministic: an upgrade is derived from a legacy deposit that every node sees and can construct for itself, and a redemption burn is permissionless, so no participant can keep from others the weight such a transaction carries. They are ordered: upgrades are minted in the order their deposits appear on the legacy chain, so they cannot be reordered, held back, or packed into a block of an adversary's choosing. And an upgrade's share decays with the delay between its legacy confirmation and its inclusion, by the factor ρ_j of Eq. 20, so the largest part of it goes to whoever includes it first. What remains is that a branch carrying bridge transactions outweighs one that does not, which is the incentive the term exists to create. The lever also exists only while bridge transactions are still being produced, a bounded interval rather than an open-ended one: the bridge closes at T_{close} (Section 4.1.2), after which no block can carry one and $W_{\text{bridge}}(h) = 0$ for every block by Eq. 20. Blocks weighted before the close keep that weight in cumulative weight, but it cannot be exploited: two competing branches share their history up to the fork point, so the bridge weight in that shared prefix enters both sums identically and cancels, leaving only the differing suffixes to decide. Past T_{close} no block in either suffix can carry bridge weight, so created weight can no longer distinguish one branch from another.

To mitigate the risk of deep block reorganizations and malicious history rewrites, the fork-choice rule incorporates a dynamic penalty multiplier applied to competing forks.

Consider a scenario where a proposed fork (C_{fork}) attempts to replace a sequence of blocks from the canonical chain (C_{current}). The condition for the network to adopt a competing fork is defined by Eq. 22, with the threshold required to overwrite history expressed as a function of the reorganization's "effective depth," d_{eff}:

$$W_{\text{chain}}(C_{\text{fork}}) > W_{\text{chain}}(C_{\text{current}}) + \big(P(d_{\text{eff}}) - 1\big) \cdot \Delta W_{\text{stake}}(C_{\text{current}}) \tag{22}$$

where $\Delta W_{\text{stake}}(C_{\text{current}})$ is the stake weight the canonical branch has accumulated since the fork point, with its bridge-derived weight excluded. The penalty is additive on the disputed suffix rather than multiplicative on the whole of chain weight: what a fork must overcome is the weight of the history it proposes to replace, raised by the penalty factor, not the accumulated weight of the chain back to genesis. Excluding the bridge contribution keeps protocol-created weight from inflating the barrier as well as the branch.

The parameter d_{eff} is the number of timeslots between the fork point and the tip, less a propagation buffer of 100 timeslots — the same constant as the heartbeat interval (Table 2). Measuring in elapsed timeslots rather than in blocks means a branch that skipped slots is treated as though it were correspondingly deeper. A reorganization is accepted without penalty when $d_{\text{eff}} \leq 0$, and also when the two branches differ in height by fewer than six blocks. The penalty function $P(d_{\text{eff}})$ is defined in Eq. 23 and follows a deterministic, everywhere-continuous curve (see Figure 5):

$$P(d_{\text{eff}}) = \begin{cases} 1 & \text{if } d_{\text{eff}} \leq 0 \\ d_{\text{eff}}/8 + 1 & \text{if } 0 < d_{\text{eff}} < 248 \\ 32 & \text{if } 248 \leq d_{\text{eff}} < 900 \\ 960/\sqrt{d_{\text{eff}}} & \text{if } 900 \leq d_{\text{eff}} < 921600 \\ 1 & \text{if } d_{\text{eff}} \geq 921600 \end{cases} \tag{23}$$

The logic of the phases ensures both immediate finality and long-term network convergence:

- **Propagation Buffer ($\leq$ 100 timeslots):** Reorganizations reaching no further back than the buffer pass unpenalized, accommodating natural network latency and propagation delays without penalizing honest validators. This is a subtraction from the measured depth rather than a segment of the curve.

- **Linear Ramp (101 to 347 timeslots of total depth):** As reorganization depth increases, the required weight scales linearly, reaching the ceiling exactly at $d_{\text{eff}} = 248$ — a fork reaching some 58 minutes back must present suffix weight 32 times that of the history it replaces.

- **Plateau Phase (348 to 999 timeslots):** The penalty holds at its ceiling of 32, ensuring that medium-term history (up to $\sim$2.8 hours) is practically immutable under normal operating conditions.

- **Decay Phase (1,000 to 921,699 timeslots):** To prevent permanent network partitions, a decaying penalty governed by an inverse-square-root curve ($1/\sqrt{d_{\text{eff}}}$) allows isolated network fragments to eventually reconverge. A partition of $\sim$1 week requires only a 4x weight multiplier.

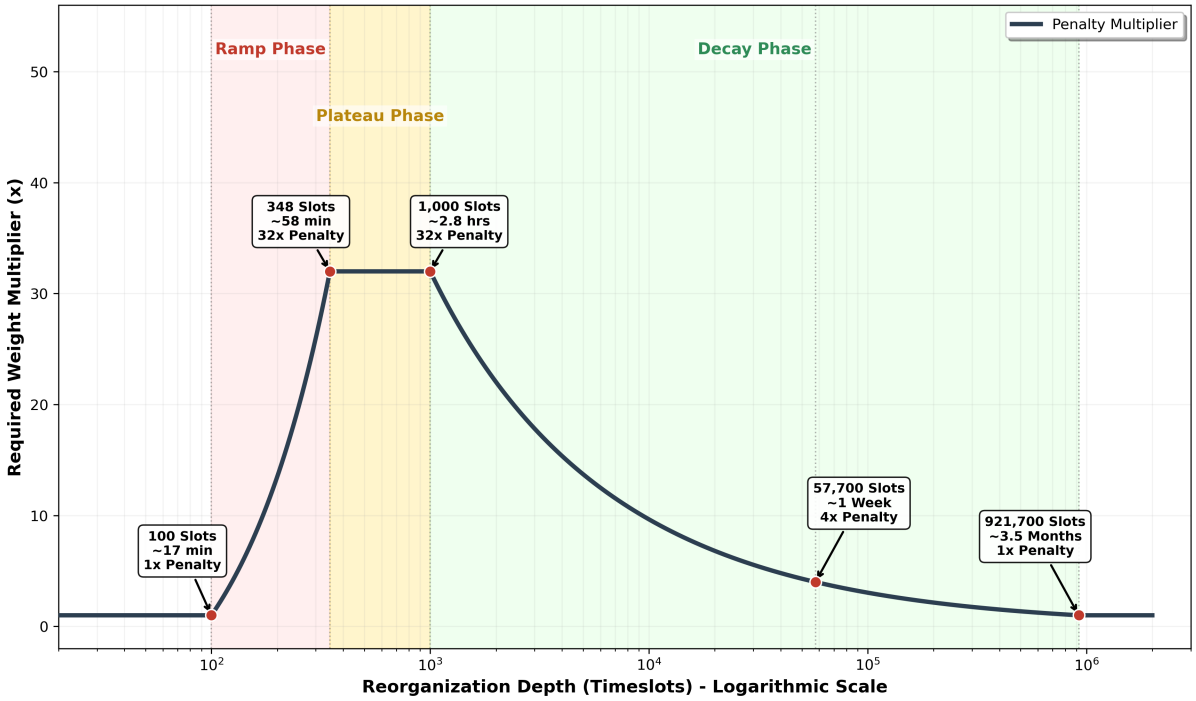


Figure 5: The Reorganization Penalty Curve. Short-term history is heavily protected (requiring up to 32x relative weight to reorganize), while long-term network partitions are permitted to heal via a decaying penalty constraint.

- **Normalization ($d_{\text{eff}} \geq 921{,}600$):** After ∼106 days, the penalty multiplier resolves back to 1, and historical convergence relies purely on baseline cumulative chain weight (Table 2).

A second constraint sits ahead of the weight rule. The client carries consensus checkpoints: block hashes fixed at specific heights, which any block at that height must match and below which no fork is considered at all. That check runs before the weight comparison of Eq. 22 and decides on height and hash alone, so a reorganization forking at or below the most recent checkpoint is rejected however heavy its branch, and the Decay and Normalization phases are reachable only for forks above it.

5.6 Slashing: Penalizing Equivocation

In a Proof-of-Stake chain, producing a block signature carries no intrinsic resource cost. Absent countermeasures, a rational validator is therefore tempted to sign on every competing branch at once (the "nothing-at-stake" problem), since doing so maximizes the chance of being on the winning chain at no cost. Such behavior undermines convergence and enables cheap short-range reorganizations. qBitcoin removes the temptation by making *equivocation* — endorsing two conflicting histories with the same coins — a punishable, on-chain offense.

5.6.1 Equivocation and Its Proof

As defined in Section 5.1, a stake signature is bound to a specific (staked output, timeslot) pair. Equivocation is defined as a single staked output authorizing two different blocks in the same timeslot. Since the two blocks differ (different parent and/or different transaction set), the

two signatures are computed over different messages, yet both are produced by the same key spending the same output in the same timeslot. The pair of signatures is thus a self-contained cryptographic proof that the validator equivocated — verifiable by anyone and requiring no trust in the reporter.

5.6.2 The Slashing Transaction

A dedicated transaction type, `TX_TYPE_SLASHING`, carries this proof on-chain. The transaction is trustless and permissionless: it bears no signature of its own; any participant who observes two conflicting stake signatures can construct it, and every participant derives a byte-for-byte identical transaction from the same evidence, as its form is fully deterministic. The slashing transaction:

- embeds the two conflicting stake signatures as evidence;
- spends the equivocated staked outputs — the proof of equivocation, rather than an owner signature, is what authorizes the spend;
- returns to each penalized owner its balance minus a fine of the fixed fraction ϕ_{slash} (Table 2), which is routed in its entirety to the Network Reward Fund (Section 4.2) on the same footing as transaction fees, entering the block inflow Φ_h of Eq. 8;
- produces refund outputs subject to the same maturity lock that ordinary stake outputs carry.

A given staked output is consumed by either a normal stake transaction or a slashing transaction: the two are mutually exclusive spends of the same coins.

5.6.3 Branch Invalidation

The decisive consensus rule is invalidity rather than weight: once a slashing transaction for a staked output exists, the equivocating stake transaction that spent that output is invalid, and therefore every block built upon it is invalid, regardless of the cumulative weight its branch has accumulated. Validity screening precedes the weight comparison of Eq. 22: the fork-choice rule of Section 5.5 selects the heaviest *valid* chain, so an invalidated branch stays out of contention however much stake weight it goes on to gather.

Honest nodes that hold the evidence reject such a branch and adopt the branch that carries the slashing transaction. As the slashing transaction propagates, observation of the evidence — and hence rejection of the equivocating branch — spreads across the network, and the chain converges on the punished outcome. The penalty is therefore robust against an equivocator simply piling additional weight onto its own branch.

5.6.4 Weight Accounting and Inclusion Incentive

So that the punishing branch keeps the weight it is due, the outputs consumed by a slashing transaction contribute their full age-weighted stake weight to the block that includes it — exactly the weight they would have contributed as stake, computed by the value $\times$ age rule of Eq. 19. That contribution is the term $W_{\text{slash}}(h)$ of Eq. 20.

The term $W_{\text{slash}}(h)$ is therefore the aggregate value $\times$ age weight of all outputs slashed in block h. The branch that slashes an equivocator thus inherits the weight the equivocator's stake would have carried. This weight contribution is also the validator's incentive: although the

fine itself flows entirely to the Reward Fund, including a slashing transaction directly increases the block's chance of winning its timeslot — and with it, the block reward of Eq. 10. Only slashing evidence and bridge transactions carry weight that responds to what the transaction is; an ordinary transfer contributes the single unit N_h counts for every transaction alike, and the size of its fee contributes nothing.

W_{slash} is *conserved*: it is exactly the weight the equivocator's own stake would have contributed, so counting it moves weight between branches rather than creating any. W_{bridge} is created by the protocol instead; Section 5.5 sets out the consequence for cumulative weight.

5.6.5 Scope and Limits

Slashing is effective for equivocation discovered within the evidence retention horizon, which is 106 blocks: the propagation buffer of Section 5.5 plus the six in-core levels, and so the depth at which the reorganization penalty first begins to bite. At full cadence that is under twenty minutes. It is a distinct quantity from the penalty's normalization horizon, four orders of magnitude further out: slashing reaches back as far as the retention horizon, and defense against equivocation older than that rests on weak subjectivity and on the consensus checkpoints described in Section 5.5. The fine fraction and the retention horizon are fixed protocol constants (Table 2), ensuring that slashing transactions remain byte-for-byte deterministic across all nodes. One class of output lies outside slashing's reach altogether: the genesis stake of Section 5.2, whose balance may never decrease and from which no fine can therefore be taken. Where genesis outputs alone equivocate, no slashing transaction is constructed, and neither a fine nor a ban follows.

6 Cryptographic and Architectural Upgrades

With the economic foundation and consensus mechanisms established, the qBitcoin architecture introduces several key cryptographic innovations. These upgrades enhance robustness, privacy, and efficiency, including a simplified P2SH-only address system, the integration of lattice-based post-quantum cryptography, and enhanced script privacy through MAST.

6.1 Simplified Address System (P2SH-Only)

To eliminate the complexity associated with multiple address types (e.g., P2PKH, P2SH, SegWit), the upgraded protocol will exclusively utilize a Pay-to-Script-Hash (P2SH) model. The transaction output commits to the hash of the script, while the input provides the pre-image script itself alongside the necessary signatures. The single output form simplifies protocol integration for wallets and developers. Transaction malleability remains a property of the serialization: the transaction hash is taken over the full transaction, signatures included, so a change in signature encoding changes the identifier of an otherwise identical transaction.

One rule of the legacy standard does not carry over. Bitcoin caps every element a script may push onto the stack at 520 bytes, a limit no post-quantum scheme can respect: a Falcon-512 public key alone is 897 bytes, and its signatures also exceed the cap. qBitcoin therefore drops the per-element limit and bounds the redeem script and the signature list at 65,535 bytes each (Table 2). This is a necessary condition for the signatures of Section 6.2 rather than a discretionary change, and it is the only respect in which the model departs from P2SH: the output form, the address form and the spend-by-preimage rule are unchanged.

6.2 Post-Quantum and Crypto-Agile Signatures

The advent of quantum computers poses a threat to cryptocurrencies relying solely on elliptic curve cryptography (ECDSA) [6]. Specifically, Shor's algorithm renders legacy secp256k1 signatures vulnerable to future private key derivation. To address the quantum threat while preserving interoperability during migration, qBitcoin adopts a crypto-agile signature framework supporting quantum-resilient Falcon alongside classical Schnorr and ECDSA.

Falcon [6], selected for standardization by the U.S. National Institute of Standards and Technology (NIST) [5], provides the post-quantum signature path. While all post-quantum signature schemes inherently require larger key and signature sizes than classical ECDSA or Schnorr signatures, Falcon is well-suited for blockchain applications compared to other NIST finalists (such as Dilithium/ML-DSA or SPHINCS+). Its security is based on the hardness of the Short Integer Solution (SIS) problem over NTRU lattices, which allows it to generate the most compact digital signatures among lattice-based schemes, paired with fast verification times.

Under the crypto-agile model, Falcon provides long-term post-quantum security, while Schnorr and ECDSA remain available for compatibility, migration, and policy-defined transaction classes. This lets the protocol evolve its cryptographic requirements without forcing a disruptive address or transaction-format break.

By integrating crypto-agile signature support, qBitcoin achieves the following:

- **Long-Term Security:** The chain is proactively secured against future threats from quantum adversaries, protecting the integrity of the supply schedule and user funds indefinitely.
- **Optimized Scalability Trade-offs:** While PQC introduces a larger data footprint than legacy ECDSA, Falcon minimizes the inevitable block size increase better than alternative quantum-resistant schemes. Furthermore, Falcon's rapid verification speeds often outperform classical ECDSA verification, offsetting bandwidth costs by significantly reducing CPU overhead during block validation.
- **Standardization and Auditability:** Relying on a NIST-selected algorithm ensures that the cryptographic primitives have undergone extensive public and academic scrutiny, lending high confidence to their security and implementation.

6.3 Enhanced Script Privacy with MAST

To further enhance user privacy, minimize the on-chain data footprint, and reduce public key exposure to quantum threats, qBitcoin integrates a mechanism inspired by MAST [8], proposed for Bitcoin but rejected and never activated there, and by the conceptual goals of Pay-to-Merkle-Root (BIP 360) [9]. Crucially, rather than introducing an entirely new address type, qBitcoin embeds the MAST capability directly within the existing Pay-to-Script-Hash (P2SH) standard.

The UTXO still simply pays to a standard script hash, but the introduction of new opcodes allows the underlying script to natively support Merkle path and target script verification. The extended script model allows a UTXO to be locked to the Merkle root of a tree containing multiple alternative spending conditions.

When the UTXO is spent, only the script that is actually used and its corresponding Merkle path are revealed on-chain. The existence of any other potential spending paths remains private and does not consume block space. For instance, in a 3-of-5 multisignature setup, the on-chain transaction would not need to reveal all five potential public keys. If three participants

cooperate to sign, only their specific 3-of-3 script path is revealed, completely obscuring the fact that two other signers were also possible.

MAST capability is enabled through two new opcodes, `OP_MASTVERIFY` and `OP_EXEC`, designed to work within a standard P2SH Redeem Script. They are independent primitives rather than a fixed pair: the first proves that a committed leaf belongs to the tree, the second executes a script taken from the stack. The minimal redeem script that proves membership is `<MerkleRoot> OP_MASTVERIFY OP_1`, with the spender supplying the leaf hash and the Merkle path in the input; the trailing `OP_1` supplies the true value the script terminates on, since `OP_MASTVERIFY` consumes its operands and leaves the stack empty.

- **`OP_MASTVERIFY`:**
  - **Operation:** Verifies that a committed leaf is a member of a Merkle tree rooted at a specific hash.
  - **Stack Transition:**
    `<LeafHash> <MerklePath> <MerkleRoot>` $\rightarrow$ (nothing)
  - **Logic:** The opcode takes a 32-byte `LeafHash`, walks the `MerklePath` to recompute a root, and verifies that it matches the `MerkleRoot` provided on the stack. All three operands are consumed and no value is pushed; if the check fails, the transaction is immediately invalidated. The path is a bare concatenation of 32-byte sibling hashes carrying no direction bits: each step combines the running hash with its sibling in sorted order, so the tree must be built the same way.
- **`OP_EXEC`:**
  - **Operation:** Treats the data on top of the stack as a script and executes it.
  - **Stack Transition:**
    `<LeafScript>` $\rightarrow$ `<Result>`
  - **Logic:** The top stack item is consumed, deserialized as executable script code, and run within the current context. The execution context allows the arguments (e.g., signatures) placed on the stack before the script to be consumed by the verified logic.

The spending process is therefore compact and deterministic: the spender provides the necessary signatures, the Merkle proof, and the leaf it claims. Membership is verified before the branch runs. Because `OP_MASTVERIFY` commits to a leaf *hash* rather than to the leaf script itself, a redeem script that both proves membership and runs the proven branch is responsible for tying the two together: the executed script must be the pre-image of the hash the proof verified, and the redeem script's construction is what enforces it. The MAST integration brings advanced privacy features directly into the scripting layer without requiring complex new transaction versioning or fracturing the ecosystem with new address formats.

6.4 Client Implementation and Interoperability

To ensure immediate viability and developer adoption, the qBitcoin core node software is fully open-source. The node exposes a Remote Procedure Call (RPC) API highly compatible with the legacy Bitcoin Core standard, allowing exchanges, block explorers, and existing infrastructure providers to integrate qBitcoin with minimal friction. Furthermore, the node software is deployable via standard containerization (Docker), ensuring cross-platform stability.

7 Extended Capabilities: Client-Side Smart Contracts

7.1 Design Philosophy: A Hybrid Validation Model

The proposed architecture is designed to support a highly scalable smart contract ecosystem by employing a hybrid validation model. The hybrid model distinguishes between two types of digital assets to balance efficiency with flexibility:

Tier 1: Native Embedded Tokens. For standard digital assets, the protocol integrates validation directly into the consensus layer. The node software recognizes a specific transaction type designated as "tokens". These transactions are exceptionally lightweight and do not require the node to maintain a global virtual machine state (such as in Ethereum), enabling full validation on standard consumer hardware, including mobile devices.

To maintain maximum verification speed, the protocol mandates that a single transaction handles only one token type. A native token transaction includes a `token_id` attribute, which is the hash of the transaction that created the contract. If the `token_id` is empty, the transaction is inherently recognized as the genesis transaction for a new asset.

Outputs for "tokens" transactions are strictly categorized into two types, determined by the first byte of their data field:

- **Transfer Outputs:** Contain the transferred token amount as a 64-bit integer.
- **Control Outputs:** Contain a set of control attributes ordered by their attribute ID. These include a `permissions` bitmask (e.g., the "mint" bit) and metadata fields such as `decimals`, `name`, and `symbol`. Crucially, the `decimals`, `name`, and `symbol` attributes are exclusively permitted in the genesis transaction.

To ensure forward compatibility without requiring a network hard-fork, any unknown attributes included in a control output are intentionally ignored by the node.

While native tokens eschew Turing-completeness, the omission is a distinct architectural advantage rather than a limitation. By relying on the foundational Bitcoin scripting system, advanced features like multi-signature setups and timelocks are natively inherited. Protecting token outputs using standard UTXO cryptography is inherently more secure and significantly easier to implement than coding equivalent protections in account-based smart contract platforms like Ethereum or Tron.

Furthermore, the UTXO model offers exceptional fee flexibility. Transaction fees do not need to be paid by the token sender. A transaction can include a separate input (e.g., from a service provider) to cover the network fee. Fee-input separation simplifies user onboarding by enabling "sponsored" transactions without requiring users to calculate unpredictable `gas_limit` parameters.

Tier 2: Client-Side Execution. For cases where native embedded tokens are insufficient and more complex, arbitrary algorithms are required (such as Solidity-style contracts or RGB protocols), the model shifts the burden of execution to external applications. Every transaction output includes an additional `data` field designed to carry arbitrary information. The qBitcoin node transmits the `data` field without any validation or interpretation.

The execution of the smart contract logic and the maintenance of token balances occur entirely on the client side, managed strictly by the users and applications participating in those specific contracts. To facilitate efficient indexing by external applications, it is recommended that the first output of such transactions be directed to a specific contract address, making it trivial to distinguish the transactions of one contract from another. The client-side execution

model prevents blockchain bloat and network-wide computational overhead. An example of a multi-party UTXO transaction is illustrated in Figure 6.

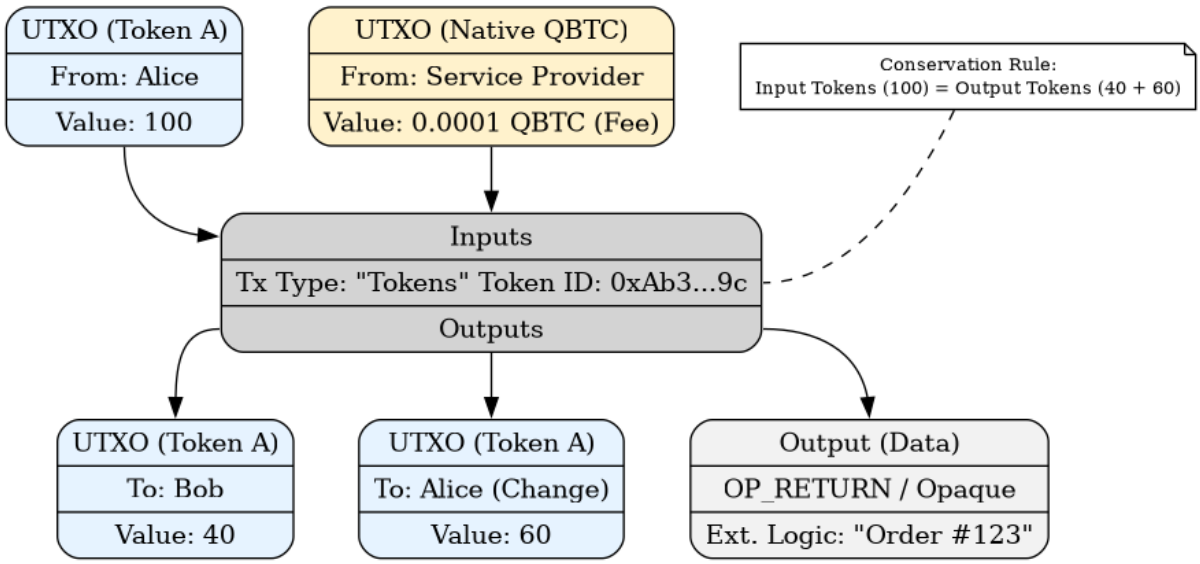


Figure 6: An example of a complex transaction in the proposed qBitcoin protocol. The existing UTXO model can be leveraged to atomically handle multiple parties and logic for client-side smart contracts, using script hashes for contract interactions.

7.2 UTXO-based Implementation Concept

The UTXO model is exceptionally well-suited for the hybrid validation approach. Unlike account-based models, there is no need to calculate a "gas limit" or risk paying fees for failed executions. Fees are deterministic, and transactions are binary: they are either valid or invalid.

For Decentralized Exchange (DEX), the protocol utilizes Atomic Swaps across separate transactions rather than complex single-transaction logic. Since a transaction carries only one token type to ensure lightweight validation, a swap is executed via cryptographically linked transactions on the same chain:

- **Transaction A:** User 1 transfers Token A to User 2.
- **Transaction B:** User 2 transfers Token B to User 1.

In the atomic swap scenario, the transactions are linked (for example, via hash time-locks) such that one cannot be finalized without the other. The linked structure facilitates trustless exchange without the overhead of a global virtual machine or the congestion of shared liquidity pools.

7.3 Architectural References and Design Lineage

The qBitcoin smart contract model consciously avoids the non-deterministic execution costs and state bloat associated with the "World Computer" (EVM) model. Instead, it synthesizes proven high-efficiency mechanisms from the UTXO-based protocol lineage:

- **The "Minting Baton" Mechanism (SLP):** To manage token authority without centralized contract state, qBitcoin adopts the "Minting Baton" concept [10]. Administrative privileges are represented as a "Control" output containing a permission bitmask with the "mint" bit enabled. The consensus rules strictly dictate that a token transaction's output sum cannot exceed its input sum unless it is the genesis transaction or it consumes an input with the "mint" permission. If a transaction consumes a "mint" input but does not produce a corresponding "mint" output, the ability to create new tokens for that `token_id` is permanently destroyed.

- **Native Asset Validation (Ravencoin Model):** Unlike early "Colored Coin" implementations which relied solely on off-chain indexers, qBitcoin enforces token conservation rules at the consensus level. Following the Ravencoin approach [11], the consensus engine validates the balance of inputs and outputs natively. The first-class consensus status for tokens enables light clients to verify token validity without trusting a third-party indexer to detect inflation bugs.

- **Client-Side Validation (RGB and Taproot Assets):** For complex applications (Tier 2), the protocol adopts the Client-Side Validation paradigm proposed by Peter Todd [12] and implemented in the RGB Protocol [13]. By using the blockchain strictly as a data availability layer via the arbitrary `data` field, while shifting execution to the client, qBitcoin achieves massive scalability.

8 Roadmap: The Evolution Path

8.1 Phase 1: The Distribution (2009-2025)

The qBitcoin project follows a phased development plan that builds upon the solid foundation of the Bitcoin legacy. The evolutionary path, transitioning from the distribution phase to the stewardship phase, is illustrated in Figure 7.

- **Bitcoin Launch & Initial Issuance:** Launch of the Bitcoin protocol in 2009. Establishment of the 21 million supply cap and the UTXO model.

- **PoW Maturity Reached:** By 2025, the primary distribution phase is functionally complete, coinciding with the recognition of the impending Quantum Threat and the need for a non-destructive upgrade path.

8.2 Phase 2: The Evolution (2025-2026)

- **Stewardship Upgrade & PQC Integration:** Development and implementation of the qBitcoin node software, integrating the multi-scheme signature engine and the bidirectional bridge.

- **PoS Implementation:** The Genesis Upgrade launches the qBitcoin Proof-of-Stake chain on 2026-09-13 11:46:40 UTC, opening the initial Migration Window at a 1:1 conversion rate. Every interval in this paper is measured from that instant, the halving eras of Section 4.2.4 included.

8.3 Phase 3: The Transition (2026-2027)

- **Active Migration & Falling Conversion Rate:** Every 4,200 BTC upgraded lowers the conversion rate by 0.1%. Later holders receive fewer QBTC per BTC, so QBTC upgraded early becomes exchangeable for more BTC, and users are economically motivated to migrate early and secure their assets on the post-quantum chain.
- **Fund Seeding:** A flat 1% of every bridge mint is automatically routed to capitalize the validator Reward Fund.

8.4 Phase 4: The Utility and Stewardship (2027+)

- **Conversion Close & Emission Activation:** At T_{close}, at most 10 years after the Genesis Upgrade and sooner if the upgrade cap of 10,500,000 BTC fills or an output of the veto set moves, the bridge closes in both directions and the bridged supply becomes final. The terminal emission schedule activates on the same event, succeeding the conversion fee as the network's non-fee income and delivering at most 4 Million QBTC, paid to block proposers, before halving out permanently.
- **Smart Contracts & L2 Scaling Solutions:** With base layer security established and the majority of active supply migrated, development focus shifts to deploying Tier 2 client-side smart contracts and Layer 2 scaling infrastructure.
- **Governance Maturity:** The network achieves stable, long-term PoS security. Validators are sustained by the Reward Fund, into which all transaction fees are routed, together with the terminal emission paid directly to proposers, without reliance on PoW-based issuance, rendering the legacy chain functionally archived. As the emission halves out, the security budget converges on transaction fees alone.

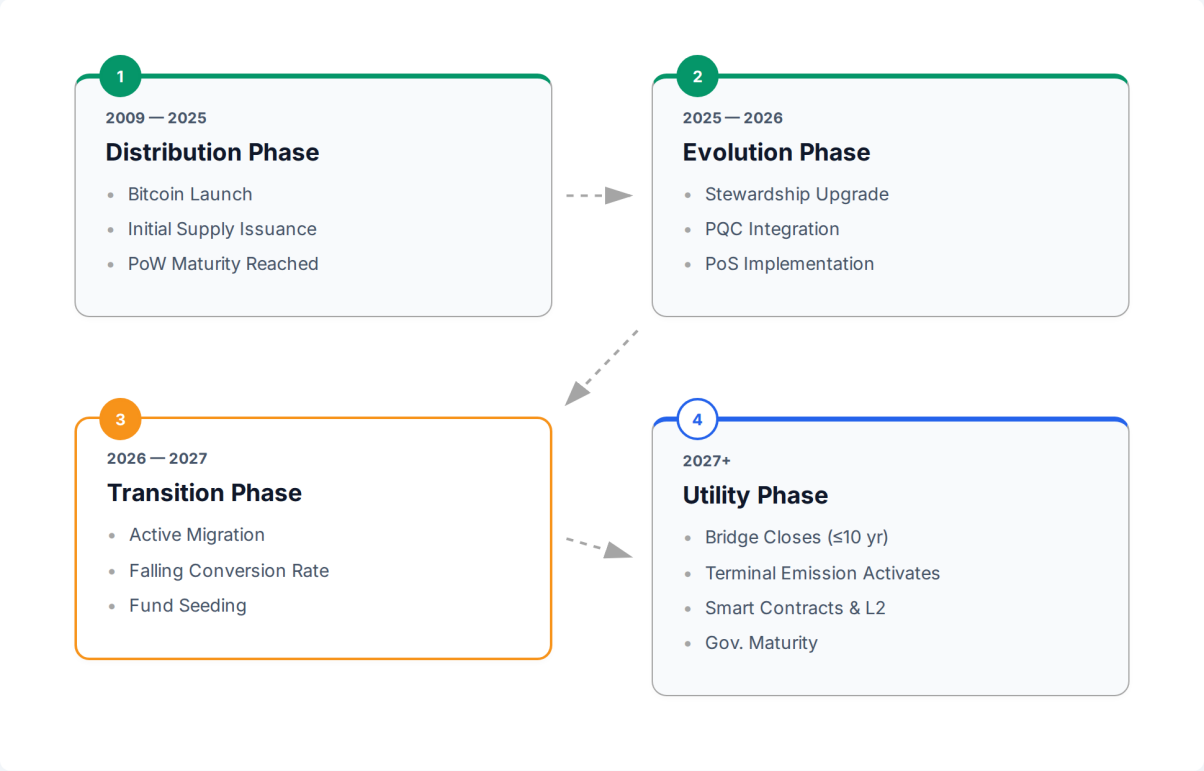


Figure 7: The evolutionary timeline of the qBitcoin protocol, transitioning from the legacy Proof-of-Work distribution phase to the Post-Quantum Stewardship era.

9 Conclusion

The Proof-of-Work mechanism successfully distributed the initial Bitcoin supply. qBitcoin upgrades the consensus layer to Proof-of-Stake to secure the distributed supply efficiently. The upgrade is done through a pegged bidirectional bridge, which allows assets to flow in both directions. The mechanism uses on-chain proofs to keep every bridged coin backed by at least one legacy BTC for the life of the bridge, and its conversion rate, which falls as upgraded volume grows, leaves part of what later holders lock unconverted, so that coins upgraded early become exchangeable for more BTC than was locked for them.

The protocol preserves the UTXO model and admits up to half of the legacy distribution base, capping the bridged supply at $\sim$3.9 Million QBTC and adding a single bounded issuance: a terminal emission of at most 4 Million QBTC that funds validators after the bridge closes and then stops, holding total supply below $\sim$7.9 Million. It integrates a crypto-agile signature engine to mitigate future cryptographic threats without sacrificing classical transaction efficiency. Furthermore, the architecture supports a scalable, hybrid validation framework for client-side smart contracts. The proposed PoS architecture ensures the technical sustainability of the network by retaining the value generated during the mining era while eliminating the dependency on energy-intensive hardware.

Disclaimer

This document is provided solely for informational and technical discussion purposes. It does not constitute financial, investment, legal, tax, accounting, or other professional advice, and it

should not be interpreted as an offer, solicitation, or recommendation to purchase, sell, or hold any asset, token, instrument, or security in any jurisdiction.

No representation or warranty is made regarding future performance, liquidity, regulatory treatment, or market adoption of the qBitcoin network or related assets. Participation in digital-asset networks involves substantial risk, including but not limited to extreme market volatility, technological failure, smart-contract or protocol defects, cybersecurity incidents, adverse regulatory action, reduced liquidity, and partial or total loss of principal.

This revision amends the monetary policy described in earlier versions of this document, in three respects that holders of legacy Bitcoin and of QBTC should read directly.

First, the bridge is limited in time and in volume, and it closes in both directions. Earlier revisions described the migration bridge without a closing date. Under this revision no legacy BTC may be locked, and no downgrade may be initiated, after T_{close}: the first of 10 years after the Genesis Upgrade, the upgrade of 10,500,000 BTC, or the spending of any output in the veto set of Section 4.1.2. The last two can occur at any time and without notice, so the window may prove far shorter than 10 years. Legacy coins not migrated by that point cannot be brought onto the qBitcoin ledger afterwards by any means. Equally, redeemability is a property of the conversion window and not a permanent feature of QBTC: a holder who does not exercise the downgrade within it retains QBTC on its own ledger but has no further claim on the legacy reserve. This risk falls hardest on dormant holders, who may take no action within the window precisely because they are inattentive to it.

Second, supply is no longer tied to the 21,000,000 legacy Bitcoin ceiling. The bridge admits at most half of it and mints at most $\sim$3.9 Million QBTC against it. The terminal emission schedule of Section 4.2.4 then mints at most 4 Million additional QBTC once the bridge has closed, paid to the proposer of each eligible block: a bounded, pre-committed dilution of at most 104% against the bridged cap, raising the ceiling on total issuance to $\sim$7.9 Million QBTC. That ceiling is reached only by a bridge closing at the Genesis Upgrade itself: the halving calendar runs from genesis whether or not the emission has begun, so the longer the conversion window lasts, the smaller the remaining schedule the close switches on (Table 8). The schedule is fixed at the Genesis Upgrade and terminates; no discretionary issuance is contemplated.

Third, the conversion rule is reversed. Earlier revisions minted one QBTC for every BTC locked, credited a share of it that grew over time to the Reward Fund, and redeemed QBTC at a flat 1:1. Under this revision the conversion rate tracks the volume standing in the reserve rather than elapsed time, falling as the bridge fills and recovering as redemptions empty it; exactly 1% of each mint is credited to the Reward Fund; nothing is minted for the part of a deposit that the lower rate leaves unconverted; and a downgrade pays that same rate read backwards, never below 1:1, so what a holder receives per QBTC depends on where the volume stands when they redeem and falls as others redeem ahead of them (Section 4.2.6).

Readers holding or evaluating QBTC on the basis of an earlier revision should read Section 4 in full.

Readers and participants are solely responsible for conducting independent due diligence and for obtaining advice from qualified professional advisers before making any financial or legal decisions related to qBitcoin, including the QBTC token.

References

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Self-published whitepaper, 2008.

[2] Sunny King and Scott Nadal. PPCoin: Peer-to-peer crypto-currency with proof-of-stake. Technical report, Self-published whitepaper, August 2012.

[3] Certicom Research. SEC 2: Recommended elliptic curve domain parameters, version 2.0. Technical report, Standards for Efficient Cryptography Group, 2010.

[4] CH Ugwuishiwu, UE Orji, CI Ugwu, and CN Asogwa. An overview of quantum cryptography and shor's algorithm. *Int. J. Adv. Trends Comput. Sci. Eng*, 9(5), 2020.

[5] National Institute of Standards and Technology. Nist announces first four quantum-resistant cryptographic algorithms, July 2022. Official Announcement. Available at https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms.

[6] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast fourier lattice-based compact signatures over NTRU, 2020. Submission to the NIST Post-Quantum Cryptography Standardization Process, Round 3. Available at https://falcon-sign.info/falcon.pdf.

[7] Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. Technical report, Blockstream, 2014. Whitepaper.

[8] Johnson Lau. BIP 114: Merkelized abstract syntax tree. https://github.com/bitcoin/bips/blob/master/bip-0114.mediawiki, 2016. Bitcoin Improvement Proposal 114, Status: Rejected. Accessed: 2026-08-24.

[9] Hunter Beast, Ethan Heilman, and Isabel Foxen Duke. BIP 360: Pay-to-merkle-root (P2MR). https://github.com/bitcoin/bips/blob/master/bip-0360.mediawiki, 2024. Bitcoin Improvement Proposal 360, Draft; assigned 2024-12-18, renamed from P2QRH to P2MR in version 0.11.0 (February 2026). Accessed: 2026-08-24.

[10] Jonald Lundeberg et al. Simple ledger protocol (SLP) specification. https://github.com/simpleledger/slp-specifications, 2018. Accessed: 2026-02-22.

[11] Tron Black, Bruce Fenton, and Joel Weight. Ravencoin: A peer-to-peer electronic system for the creation and transfer of assets. https://ravencoin.org/whitepaper/, 2018. Accessed: 2026-02-07.

[12] Peter Todd. Scalable semi-trustless asset transfer via single-use-seals and proof-of-publication. https://petertodd.org/2017/scalable-single-use-seal-asset-transfer, 2017. Accessed: 2026-03-15.

[13] LNP/BP Standards Association. RGB blackpaper: Client-side validation for Bitcoin & Lightning. Technical report, LNP/BP Standards Association, 2023. Accessed: 2026-02-07.