The BTX Post-Quantum Dossier: What Is Proven, What Is Audited, What Is Still Open

Bitcoin's remaining structural risk is the one BTX was built to remove. The first half of this document is the evidence for that sentence, layer by layer: the standards, the consensus wiring, the audit record, the network as it stands. The second answers the question that follows it: where BTX is going, and who decides what ships. The caveats are kept rather than trimmed, because they are what make the rest credible. Written for a reader who already understands Bitcoin.

SEPTEMBER 12, 2026 · 98 MIN READ · EASYBTX.COM/RESEARCH/BTX-POST-QUANTUM-DOSSIER

ABSTRACT

A friend who has held bitcoin since before its first halving asked us for a post-quantum audit of BTX. This is the answer: how BTX replaced Bitcoin's elliptic-curve signatures with the two NIST post-quantum standards from block zero, what evidence exists at every layer, what the network looks like today including the halving schedule, who holds the keys to what ships, what the 0.34.7 model network actually contains now that it is public, and the seventeen honest reasons for caution. Every claim is sourced. Every technical term is defined.

Contents

60 pages. Parts 1 to 8 answer the post-quantum question and are the shortest route through this document. Parts 9 and 10 cover the model network and who decides what ships. Part 12 is the contra column, and the glossary at the end defines every term used.

This dossier was written for one reader: a friend who has held bitcoin since before its first halving and wanted a post-quantum audit of BTX before he took the chain seriously. We did not run a new audit. We assembled the evidence that already exists, from the NIST standards down to the audit reports in our own repositories, and we kept every caveat that a reader like him would find on his own. It was researched and written by Claude Fable 5.1, an AI model working under our direction, and reviewed before publication. Every technical term is defined in the glossary at the end, and the first time a term appears it gets a short gloss in the text. Where something could not be verified, the text says so.

Revision of 14 September 2026 (V6): 0.34.7 was published for review as pull request 156, so Part 9 has been rewritten against code instead of against an announcement. The status board is redrawn and a new section covers the post-quantum transport, which is the first place BTX uses a key encapsulation mechanism on the wire rather than inside a transaction. Two corrections this document owes its readers are made in the open rather than quietly: the size figure taken from the private report was low by roughly a factor of four and a half, and the statement that the escrow shipped without the economy was true of v0.34.6 and is no longer true, because the funding path that builds, signs and broadcasts a payment is in the pull request. The five checks this article published in September are scored against what arrived, including the two it got right and the one it got wrong.

Earlier revisions, one line each. ***V5****, 14 September: a presentation pass, and the claim that no elliptic-curve verification exists anywhere in BTX script was replaced by the consensus rule that makes classical outputs impossible, which is both true and stronger.* ***V4****, 13 September: Part 10 was added, and an error this document had carried in six places was corrected, because Dandelion++ relay privacy is live from block 61,000 rather than 250,000 as the project's README still says.* ***V3****, 13 September: Part 9 was added while 0.34.7 was still an announcement.* ***V2****, 13 September: fifteen figures were added and every dated height was re-checked against the block timestamp rather than a third-party feed, which caught three wrong dates. Every number is as of 12 September 2026 unless a caption says otherwise.*

Two disclosures, because the alternative is letting you assume something friendlier than the truth. easyBTX builds the miner, the node app, PQ Wallet and qID for the BTX ecosystem. We are not the BTX core team, we hold BTX, and we take a fee from mining. Read this as a participant describing the thing it helped build, and check every claim against the sources listed at the end.

The one-page version

The claim. BTX is a fork of the Bitcoin Knots v29.2 node software that removed elliptic-curve signatures from consensus entirely and replaced them, from block zero, with the two NIST post-quantum signature standards: **ML-DSA-44** (FIPS 204, a lattice-based scheme) as the everyday key and **SLH-DSA-SHAKE-128s** (FIPS 205, a hash-based scheme) as an independent backup key. Every coin on the chain sits behind a post-quantum spend condition. There is no classical address type to migrate from, so the "burn versus steal" decision that Bitcoin will eventually face does not exist on BTX.

The money rules are Bitcoin's. A 21,000,000 cap, a block subsidy that halves (20 BTX per block, halving every 525,000 blocks), a UTXO ledger, proof-of-work, no token layer, an MIT-licensed reference node, no administrative keys and no pause switch.

The work is different. Mining is dense integer matrix multiplication, the operation that AI accelerators are built for. Since block 185,000 (10 August 2026) one mining attempt is a transformer-shaped, exactly replayable computation of about 141 trillion multiply-accumulate operations that takes roughly 30 seconds on the fastest consumer silicon. The 182-byte block header never changed.

What was announced on 13 September 2026. The lead developer described a coming release, 0.34.7, that would add a decentralized network for distributing open-weight AI models and for financing their release. On 14 September he published it as pull request 156, which anyone can now read. It is still not a release: unmerged, untagged, and the branch sets its own release flag to false. One piece of it did ship the same morning in v0.34.6, the hash-locked escrow that the financing would use, carrying an alias named `model_htlc_sha256`. Part 9 separates the two and says what the mechanism still cannot do.

Where the network stands today (12 September 2026). Block 217,689. 4,353,780 BTX mined, 20.7 percent of the cap. No halving yet; the first is expected around late July 2027 at the 90-second target. 60 reachable public nodes in 17 countries, 31 of them validating independently. One reachable mining pool. No exchange and no price.

How "post-quantum" is proven, one line per layer.

Layer	Evidence
Standards	FIPS 203, 204 and 205 finalized by NIST on 13 August 2024. ML-DSA-44 is NIST security category 2; SLH-DSA-128s is category 1.
Implementation	The node's post-quantum library vendors the NIST reference implementations (`pq-crystals/dilithium` , `sphincs/sphincsplus`) at pinned commits, with fuzz targets.
Consensus wiring	Consensus rejects any block carrying a non-OP_RETURN output that is not witness-version-2 P2MR, so the only output type is P2MR and the inherited elliptic-curve code is unreachable. Anyone can confirm this from the source and from any transaction on the explorer.
Interoperability	An independent JavaScript signer (qID) derives byte-identical addresses to the node and was cross-verified in both directions against BTX's C core for FIPS-205, pinned to NIST ACVP known-answer vectors.
Chain-level review	The core repository carries a dated audit archive: a March 2026 source audit with closeout, a June 2026 reassessment, an external NO-GO audit on the MatMul v4 fork with a remediation map, and a formal-verification suite of 21 machine-checked obligations for the shielded pool.
Our own layer	PQ Wallet, the qID identity core and the backend went through adversarial, execution-driven audits in June, July and August 2026. Verdict: no exploitable vulnerability in the wallet client or the backend; residuals documented.

The honest caveats, up front. No named third-party firm has signed off on the BTX cryptographic stack or on qID; both projects say so themselves. The network is six months old, small, and has no public market. One million coins were produced in the first afternoon into a genesis multisig. Verification of the current proof-of-work needs a GPU, so many nodes follow signed

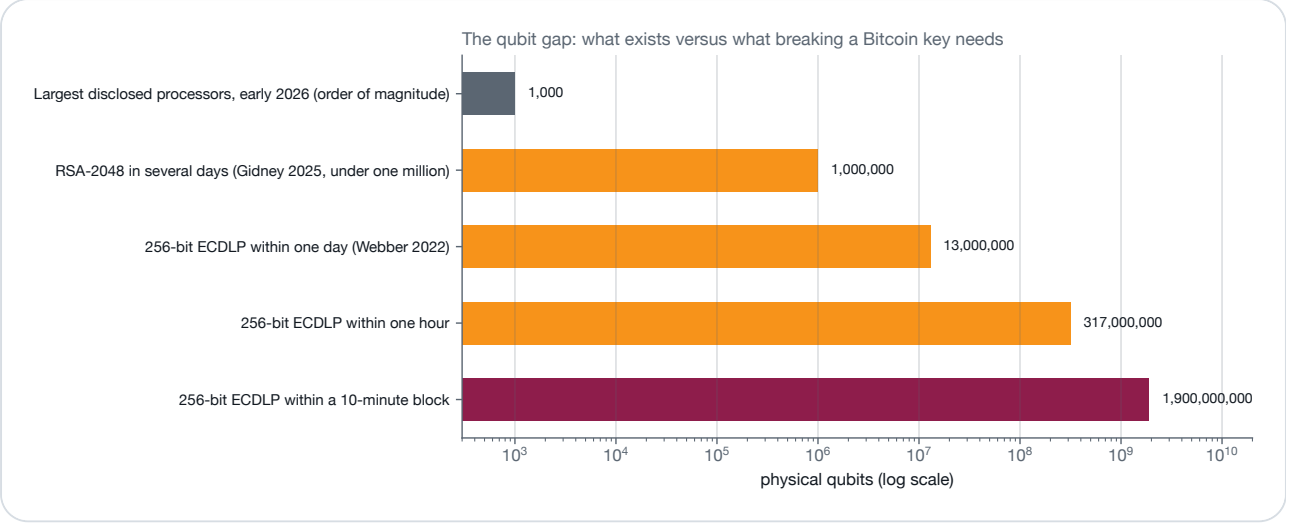
attestations today. The privacy pool the genesis block promised has been closed. The reference node has been declared handed over to community forks, and in the seventeen days after that declaration the same repository shipped 203 more commits and five more releases. One person writes and merges effectively all of it, and no pull request has ever been approved by another human being. Part 12 lists all of this without cushioning.

If you take one sentence from this document: **BTX is the only live proof-of-work chain with Bitcoin's monetary rules where the quantum migration is already finished, and nearly every remaining risk is an ordinary young-chain risk rather than a cryptographic one.** The exception is one the project published itself: pull request 137 records open consensus-level script findings, and Part 9 names them.

Part 1. The threat, in Bitcoin's own terms

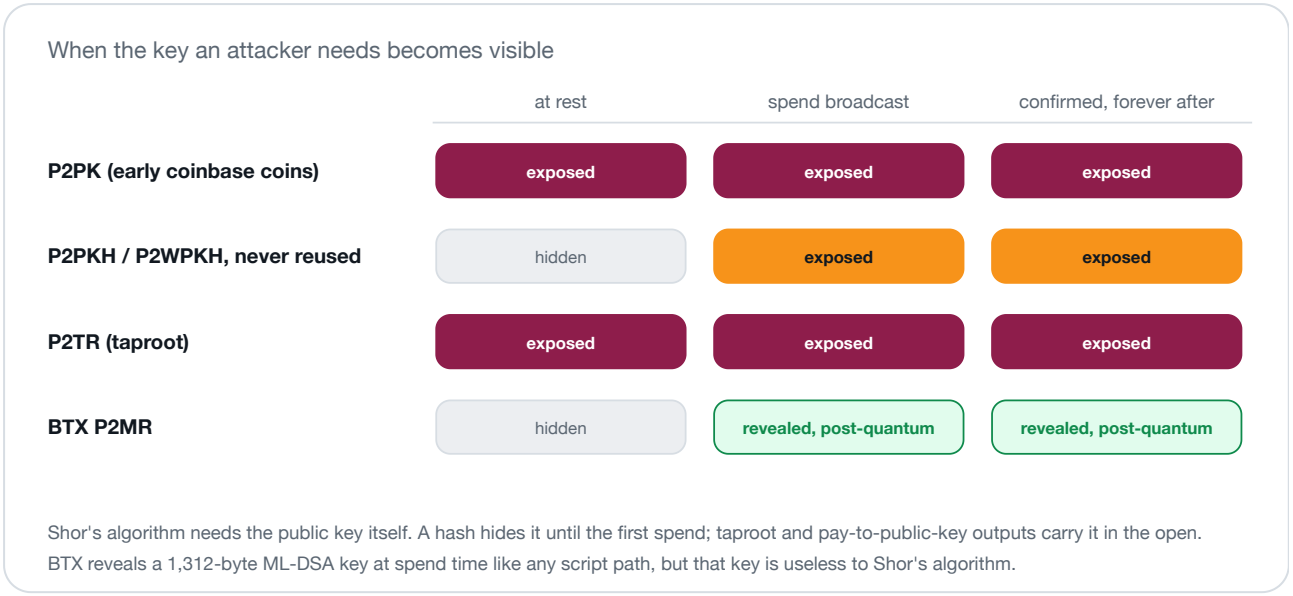
Ownership of bitcoin is proven with ECDSA (since 2009) and Schnorr signatures (BIP 340, since taproot in 2021) over the secp256k1 curve. Both rest on the elliptic-curve discrete logarithm problem (ECDLP): recovering the private scalar from the public point. Shor's algorithm, a quantum algorithm for period finding, solves that problem in polynomial time. The break is qualitative. A bigger curve does not help, and there is no patch.

Published circuit estimates put a 256-bit ECDLP at about **2,330 logical qubits** (error-corrected qubits) and about 130 billion Toffoli gates (Roetteler, Naehrig, Svore and Lauter, 2017). Translated into physical qubits under surface-code error correction (Webber and coauthors, 2022): roughly 13 million to break a key within a day, 317 million within an hour, 1.9 billion within a ten-minute block. The largest publicly disclosed processors as of early 2026 hold on the order of 1,000 physical qubits. In 2025 Gidney reduced the RSA-2048 estimate to under one million noisy qubits running for several days. The estimates are revised every few years, and they move in one direction.



The qubit gap on a logarithmic scale. Physical-qubit requirements for a 256-bit elliptic-curve discrete logarithm from Webber and coauthors (2022) at three time budgets, the RSA-2048 estimate from Gidney (2025), and the order of magnitude of the largest publicly disclosed processors as of early 2026. The bars are estimates that assume particular error rates and codes, and they are revised every few years.

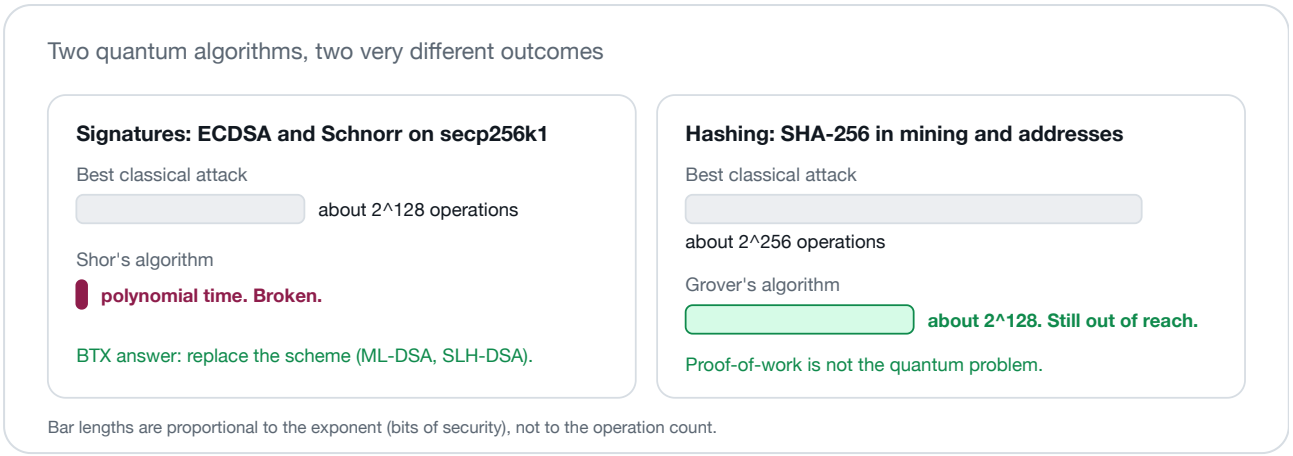
Exposure is structural. Shor's algorithm needs the public key as input, so a key is safe only while it stays hidden behind a hash. Deloitte's analysis put roughly 25 percent of all bitcoin, more than 4 million BTC, in outputs whose public keys are already visible on chain: early pay-to-public-key coinbase outputs (including the coins attributed to Satoshi Nakamoto), reused pay-to-public-key-hash addresses, and every taproot (P2TR) output, because BIP 341 places the x-only key directly in the output. Every other coin exposes its key for the length of the confirmation race whenever it moves. A blockchain is a permanent record, so exposure never expires; this is the signature-side twin of "harvest now, decrypt later".



When the key a quantum attacker needs becomes visible, by output type. Shor's algorithm needs the public key itself; a hash hides it until the first spend, while pay-to-public-key and taproot outputs carry it in the open from the start. BTX reveals its 1,312-byte ML-DSA key at spend time like any script path, and that key is useless to Shor's algorithm. Sources: BIP 341, the Deloitte exposure analysis, the BTX post-quantum specification.

The fix on Bitcoin is a proposal, not a plan. BIP 360 (pay-to-quantum-resistant-hash) is a draft. Deploying it is a soft fork, after which every holder must move coins individually. Coins whose owners are dead, absent or inattentive never move, which forces the question nobody wants: let a quantum attacker take them, or freeze them by consensus rule. As of 2026 that argument ("burn versus steal") is unresolved and active.

Mining is not the problem. Grover's algorithm gives only a quadratic speedup on SHA-256, parallelizes badly, and difficulty adjustment absorbs it. Bitcoin's quantum exposure is entirely in the signature layer. That is the layer BTX replaced.



Why the signature layer is the problem and mining is not. Shor's algorithm turns the elliptic-curve discrete logarithm from exponential to polynomial difficulty, which no parameter change can repair. Grover's algorithm only halves the exponent of a hash search, from about 2^256 to about 2^128 for SHA-256, which stays out of reach, and difficulty adjustment absorbs any gradual gain.

Governments treat the date as a deadline. NIST IR 8547 proposes deprecating quantum-vulnerable algorithms after 2030 and disallowing them after 2035. NSA's CNSA 2.0 requires national security systems to transition by 2033. Mosca's inequality (data shelf life plus migration time versus time to a cryptographically relevant quantum computer) is the arithmetic behind those dates. A bearer asset meant to be held for decades has the longest shelf life of any data class.

A note on names, because they get mangled in conversation (both "MHDSA" and "SHDSA" have been heard): the schemes are **ML-DSA** (Module-Lattice-Based Digital Signature Algorithm, formerly CRYSTALS-Dilithium) and **SLH-DSA** (Stateless Hash-Based Digital Signature Algorithm, formerly SPHINCS+). **ML-KEM** (formerly Kyber) is the matching key-encapsulation standard, FIPS 203.

Part 2. What BTX is, and what it did not change

BTX is a hard fork of the **Bitcoin Knots v29.2 node software**, not of the Bitcoin ledger. It started from its own genesis block, stamped **19 March 2026**; the first block was actually mined on **23 March 2026**. The genesis message reads:

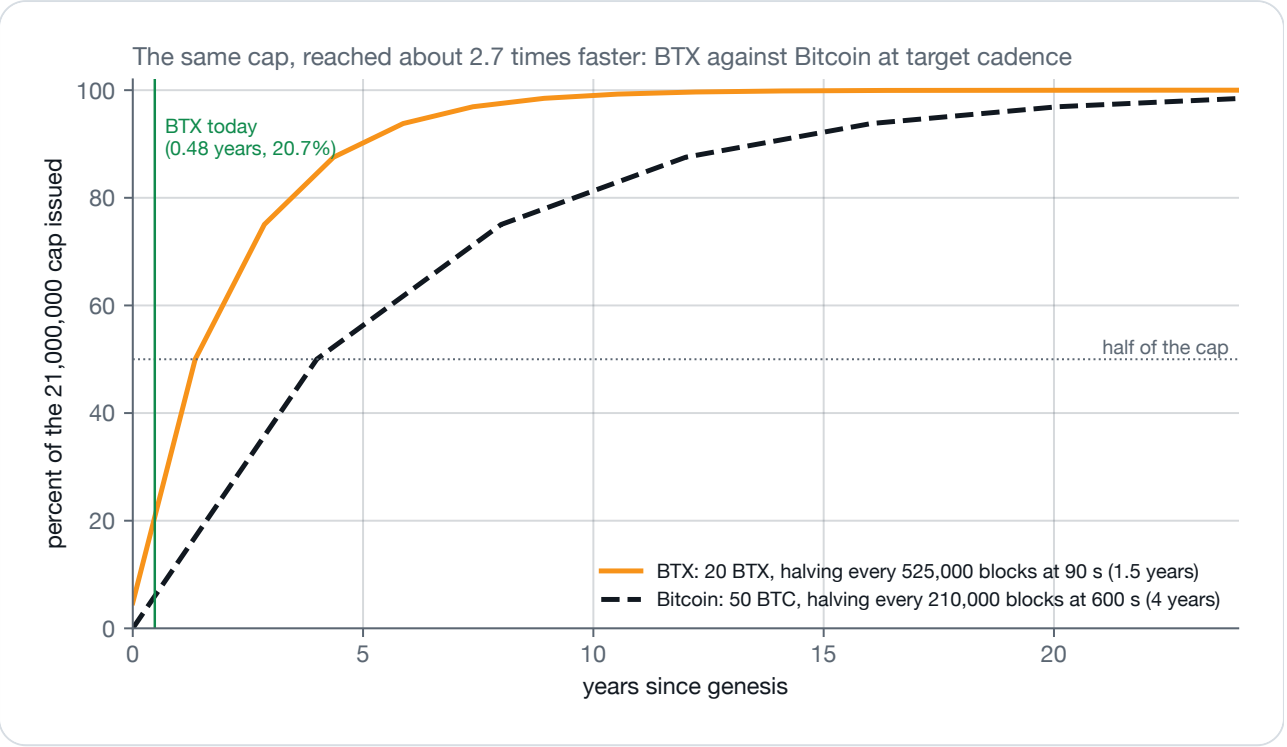
> BTX 19/Mar/2026 SMILE v2 Post-Quantum Shielded Transactions

Everything about money is inherited unchanged from the most adversarially tested financial software in existence. The new work is concentrated where it had to be.

Parameter	BTX	Bitcoin
Maximum supply	21,000,000	21,000,000
Initial block subsidy	20 BTX	50 BTC
Halving interval	every 525,000 blocks	every 210,000 blocks
Target block time	90 seconds	600 seconds
Difficulty adjustment	ASERT, every block, from genesis	every 2,016 blocks
Signatures	ML-DSA-44, SLH-DSA-SHAKE-128s	ECDSA, Schnorr

Parameter	BTX	Bitcoin
Output type	witness v2 P2MR only, addresses `btx1z…`	P2PKH, P2SH, P2WPKH, P2WSH, P2TR
Proof of work	MatMul (dense integer matrix multiplication)	SHA-256d
Block limit	24 MB, 24 million weight units	4 million weight units
Covenants	OP_CTV and OP_CSFS active from genesis	proposals
Relay privacy	Dandelion++ (live since block 61,000)	none
Token layer	none	none
License	MIT	MIT

Two things to notice. First, the halving schedule converges on the same cap: 525,000 blocks at 20 BTX, then at 10, then at 5, and so on. The chain's own test records the integer-rounding total at 20,999,999.93 BTX. Second, **there is no smart-contract or token layer**. There is only BTX. The only thing that could theoretically be counterfeited is BTX itself, and the inputs-cover-outputs rule, the range checks and the coinbase-subsidy check are Bitcoin's, enforced by every node.



The same 21,000,000 cap on two clocks. Bitcoin's 210,000-block eras at a 600-second target last about four years; BTX's 525,000-block eras at 90 seconds last about 1.5 years, so BTX reaches each milestone roughly 2.7 times sooner. The BTX curve starts at 4.8 percent because the first 50,000 blocks were mined in one afternoon; the green line is today. Both curves assume the target cadence, which real chains only approximate.

Part 3. The signature layer, precisely

3.1 P2MR: one output type, a tree of post-quantum spend paths

Every BTX output is **Pay-to-Merkle-Root (P2MR)**, witness version 2. The output script is `OP_2 <32-byte merkle root>`; the address is that root encoded in Bech32m with the `btx` prefix, which is

why every BTX address begins with `btx1z`. If you know taproot's script tree, this is the same idea with two changes: there is no key-path spend (no public key sits in the output), and the leaves verify post-quantum signatures.

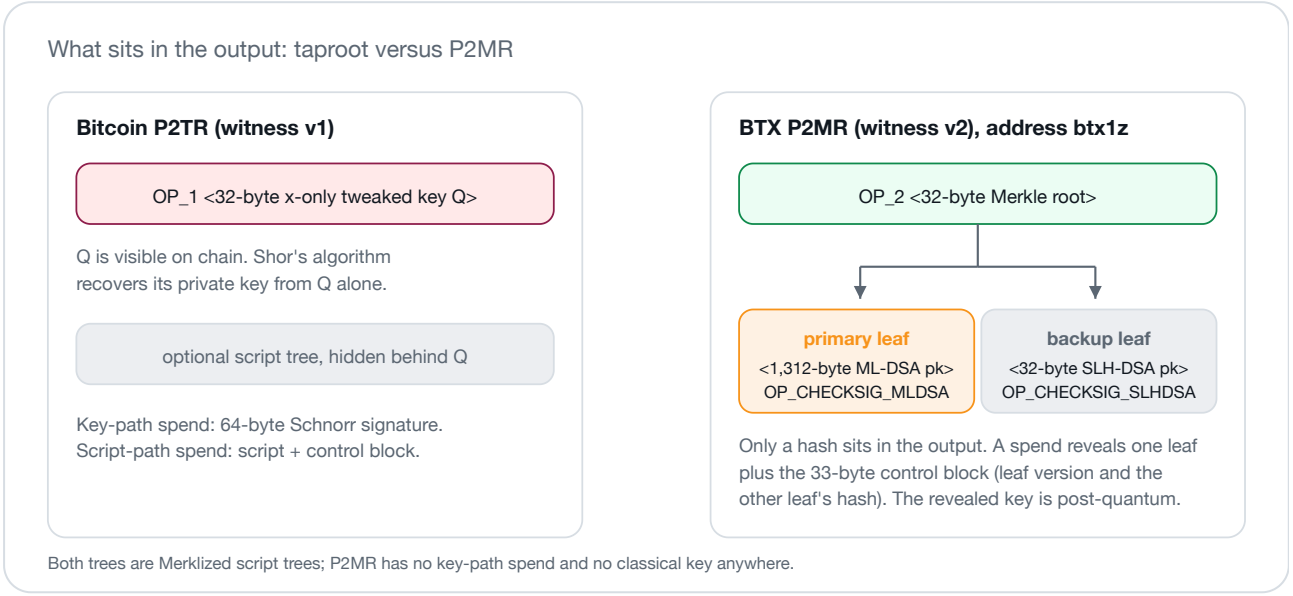
The default wallet descriptor is `mr(<ML-DSA key>, pk_slh(<SLH-DSA key>))`, which commits two leaves:

Leaf	Script	Role
Primary	`<1312-byte pubkey> OP_CHECKSIG_MLDSA`	routine spending, agent use
Backup	`<32-byte pubkey> OP_CHECKSIG_SLHDSA`	recovery path under independent assumptions

Only the branch actually used is revealed at spend time; the other stays a hash. Further leaf forms are part of consensus:

- m-of-n multisig with up to 8 keys per leaf, mixing both schemes per key (`multi_pq`, `sortedmulti_pq`), built from `OP_CHECKSIGADD_MLDSA` and `OP_CHECKSIGADD_SLHDSA`;
- timelocked multisig with `OP_CHECKLOCKTIMEVERIFY` and `OP_CHECKSEQUENCEVERIFY` (`cltv_multi_pq`, `csv_multi_pq`);
- template-constrained spends with `OP_CHECKTEMPLATEVERIFY` (`ctv_multi_pq`), which commit an output to the exact shape of the transaction allowed to spend it (vaults, payment trees);
- oracle conditions with `OP_CHECKSIGFROMSTACK`, which verifies a post-quantum signature over arbitrary stack data.

The signature hash (sighash) is BIP 341's digest structure with a distinct epoch byte for witness-version-2 execution. It commits to the input amounts, which matters for wallet safety (Part 7). Key derivation uses purpose `87h`, so hierarchical-deterministic paths look like `m/87h/…`. The Merkle leaf version is `0xc2`, and the maximum P2MR script size is 11,000 bytes.



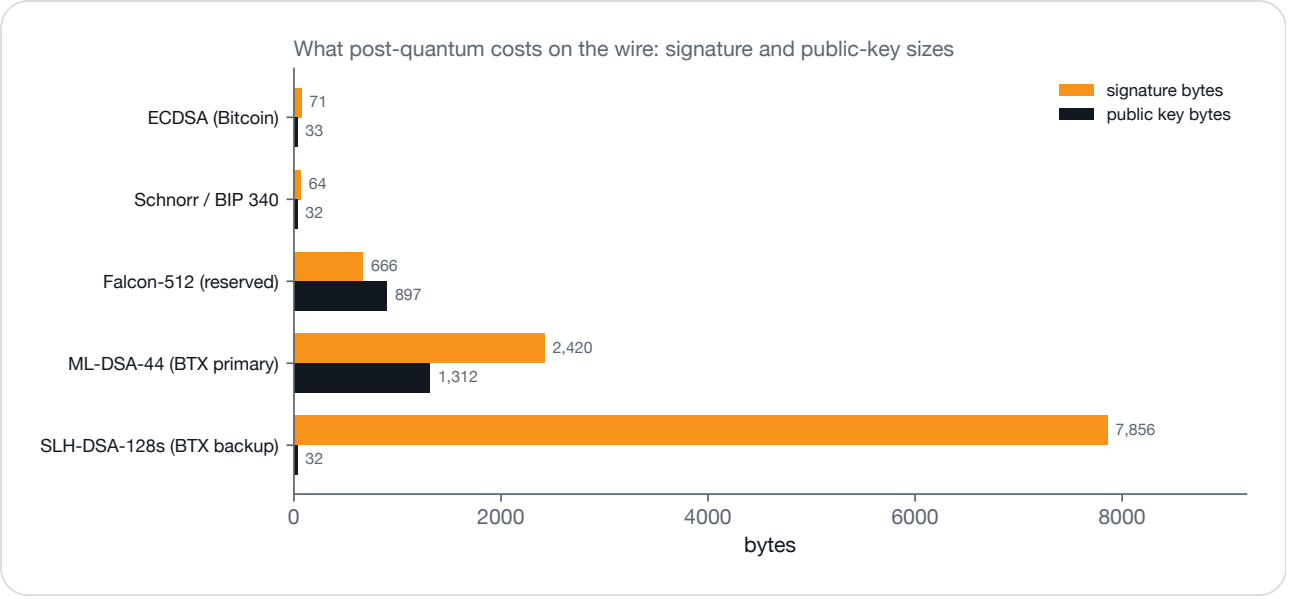
What sits in the output. A taproot output carries a 32-byte tweaked public key that a quantum attacker can work on directly. A P2MR output carries only a 32-byte Merkle root; the default tree commits an ML-DSA primary leaf and an SLH-DSA backup leaf, and a spend reveals one leaf plus a 33-byte control block holding the leaf version and the other leaf's hash. Sources: BIP 341, doc/btx-pqc-spec.md.

3.2 The two schemes, and why there are two

	ML-DSA-44	SLH-DSA-SHAKE-128s
Standard	FIPS 204 (August 2024)	FIPS 205 (August 2024)
Family	module lattice (formerly Dilithium)	stateless hash-based (formerly SPHINCS+)
Hardness assumption	Module-LWE and Module-SIS lattice problems	only the security of the hash function (SHAKE)
NIST security category	2 (collision search on SHA-256)	1 (key search on AES-128)
Public key	1,312 bytes	32 bytes
Secret key	2,560 bytes	64 bytes
Signature	2,420 bytes	7,856 bytes
Verification time versus Schnorr (Apple M1 Max)	1.94x	7.57x
Validation weight charged by consensus	1x	10x
Role in BTX	primary key	backup and cold recovery key

The pairing is the point. Lattice signatures are compact and fast, and they are what every serious deployment (TLS, SSH, Apple's iMessage PQ3, Signal) is standardizing on, but they rest on a comparatively young hardness assumption. Hash-based signatures rest only on the hash function, the most conservative assumption in cryptography, at the price of a 7.8 KB signature. BTX puts the conservative scheme into the tree as an always-present second path. If lattices fall, every BTX coin is still spendable through its SLH-DSA leaf. If hash functions fall, nothing anywhere is safe, Bitcoin included.

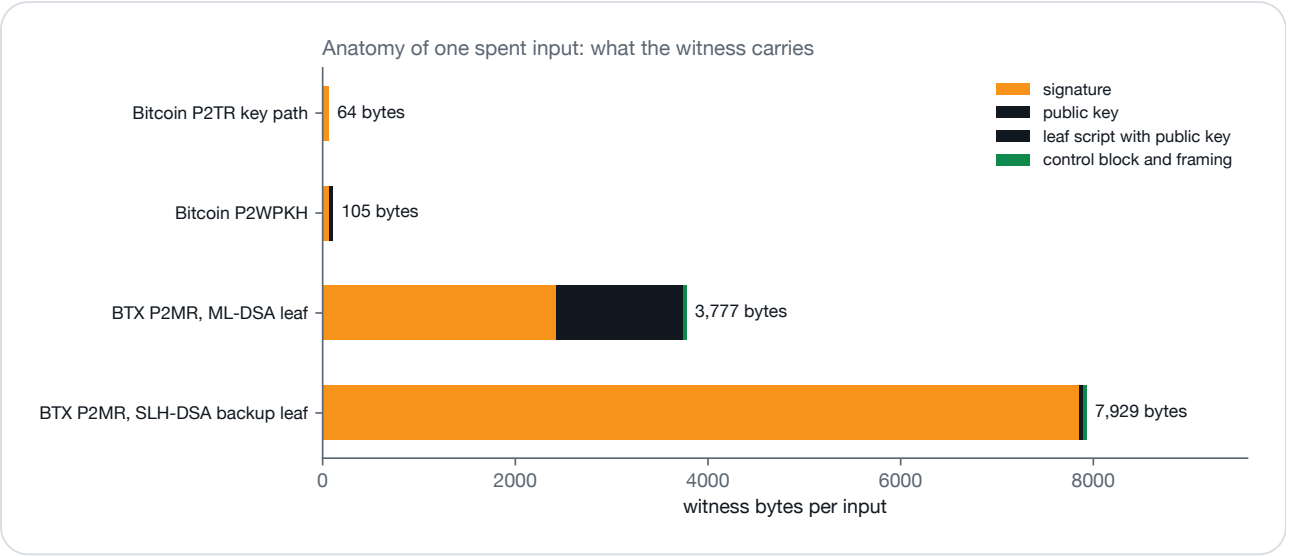
There is a third slot. Opcodes for **Falcon-512** (`OP_CHECKSIG_FALCON` and companions) are reserved but not enabled, so a smaller-signature scheme can be added by soft fork without a hard fork. That is cryptographic agility built into script, which Bitcoin does not have today.



What post-quantum costs on the wire. Signature and public-key sizes of Bitcoin's two schemes, BTX's two schemes, and the reserved Falcon-512 slot. Sources: FIPS 204, FIPS 205, BIP 340, the BTX post-quantum specification.

3.3 What it costs

Post-quantum signatures are not free, and the cost is visible on chain. Each input of an ordinary spend carries about **3.7 KB of witness data**, of which 2,420 bytes is the signature. The Bitcoin equivalent is roughly 100 bytes. The 24-million-weight-unit block limit exists to absorb that. Verification is 2x to 8x slower than Schnorr per signature; the node's own worst-case block simulation puts a P2MR block at about 2.5x the verification time of a worst-case tapscript block. These are engineering costs, disclosed, and they are the same costs BIP 360 would impose on Bitcoin.



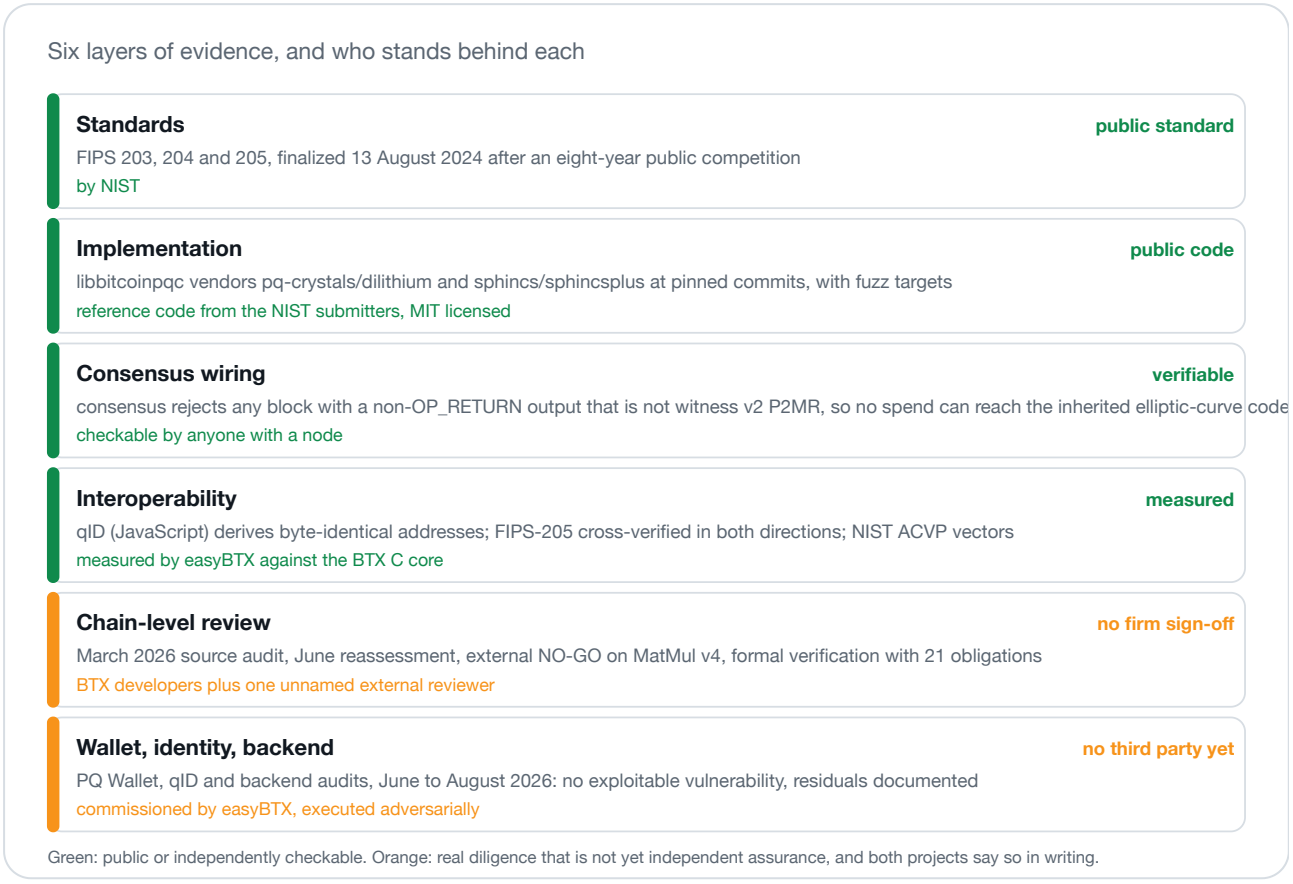
Anatomy of one spent input. The BTX ML-DSA path is reconstructed from the specification's sizes: a 2,420-byte signature, a 1,316-byte leaf script (the 1,312-byte public key with its push prefix and the checksig opcode), a 33-byte control block and 8 bytes of witness framing, 3,777 bytes in all, which is the "about 3.7 KB" in the text. It reconciles exactly with our regtest measurements: a one-input, one-output spend was 3,873 virtual bytes, and each further input added 3,818, so BTX counts witness bytes in full. The SLH-DSA backup path is 7,929 bytes. Bitcoin's P2WPKH witness is about 105 bytes and a taproot key-path spend is 64.

3.4 What is not in consensus

State this precisely, because the loose version of it is falsifiable and the precise version is stronger. BTX is a fork of Bitcoin Knots, so Bitcoin's `OP_CHECKSIG` and its elliptic-curve verification are still present in the interpreter source, and anyone who clones the repository will find them. What matters is that no output can exist whose spend would ever reach that code. Consensus forbids it: `CheckBlock` runs a rule that rejects any block containing a non-`OP_RETURN` output that is not a witness-version-2 P2MR output, and the flag that arms it is set for mainnet in the chain parameters. A transaction paying to P2PKH, P2WPKH or P2TR does not merely fail to relay. It makes the block that carries it invalid, at every height, for every node. So a classical key cannot receive BTX, cannot spend BTX, and cannot log in to a BTX application (Part 6). The inherited code is unreachable rather than absent, which is a claim you can check rather than one you have to take. "Post-quantum from genesis" means "post-quantum only", not "post-quantum available".

Part 4. How "post-quantum" is proven, layer by layer

An experienced reader asks: proven by whom?



The first four layers are public standards, public code, or measurements anyone can repeat. The last two are real diligence that is not yet independent assurance, and both BTX and easyBTX say so in writing.

4.1 The standards

NIST finalized FIPS 203 (ML-KEM), FIPS 204 (ML-DSA) and FIPS 205 (SLH-DSA) on 13 August 2024 after an eight-year public competition. BTX uses exactly the standardized parameter sets, not

the pre-standard Dilithium or SPHINCS+ variants. The node's history records the moment that mattered: early BTX validated SLH-DSA in the round-3.1 SPHINCS+ format, and the **C-002 upgrade activated FIPS-205 pure-mode verification at height 123,000**. Since then a standards-compliant FIPS-205 signer produces valid BTX recovery-leaf spends. We retested that against BTX's core at commit `f3c9eb77fa` on 6 July 2026, at the core team's request.

4.2 The implementation

The post-quantum library in the node, `src/libbitcoinpqc`, vendors the **NIST reference implementations**: `pq-crystals/dilithium` at commit `444cdcc8` and `sphincs/sphincsplus` at commit `7ec789ac`. It ships fuzz targets for key generation, sign/verify and cross-algorithm verification, and the repository's contribution rules require a local fuzz smoke run for any change under that directory. It is MIT licensed.

What it does not have: a FIPS validation certificate number (the README says the algorithms are "certified according to FIPS", which is true of the algorithms, not of this particular build), and an explicit constant-time claim. Those are the same gaps almost every open-source post-quantum deployment has today, and they are worth naming rather than glossing.

4.3 The consensus wiring

Anyone can verify that no output an elliptic-curve key could spend is permitted. `src/script/pqm.h` and `.cpp` hold the P2MR Merkle hashing and script building; `src/pqkey.h` and `.cpp` the key operations; `doc/btx-pqc-spec.md` the profile. Every transaction on btxscan.io shows a `btx1z` input and a witness that is kilobytes long. On regtest, a tampered-amount transaction is rejected by the node with the literal error `Invalid ML-DSA signature`. That line is from our own test log (section 4.6).

4.4 Interoperability: an independent signer agrees with the chain

qID, the identity and signing library inside PQ Wallet and the bonuz wallet, does not use BTX's C code. It uses the audited `@noble/post-quantum` JavaScript implementation. That gives an independent check:

- Address derivation is **byte-exact** against the node: `btxd deriveaddresses` on the canonical descriptor returns the same `btx1z…` address as the JavaScript for the test seed.
- FIPS-205 signatures were **cross-verified in both directions** against BTX's core built from the target commit (result matrix VALID, VALID, INVALID, VALID, INVALID, exactly as expected).
- Key generation is pinned to **NIST ACVP known-answer tests**, so the primitives are checked against NIST rather than against their own past output.
- Offline regression vectors pin the ML-DSA-44, SLH-DSA and ML-KEM derivations and the P2MR sighash to fixed hex, so a dependency bump that changed a byte would fail continuous integration.

Two independent codebases, in two languages, producing the same addresses and accepting each other's signatures is the strongest kind of evidence that the wiring is what the specification says.

4.5 The chain-level audit archive (BTX's own)

The core repository's `doc/` tree contains the project's security history rather than a brochure. The parts relevant here, with who performed them as stated in the documents:

Date	Document	Who	Outcome
2026-03-28	source audit, closeout in `doc/security/findings/`	internal program, tracker per finding	all mapped Critical findings closed on the reviewed path; one open item, L12 (a stronger PQ-128 shielded parameter set, moot since the pool closed)
2026-03-05	shielded-pool formal audits, rounds 1 to 6, plus a threat model	internal, with a "simulated audit report" alongside	fed the 61,000 hardening fork
2026-06-06	security audit reassessment	internal team, lattice-focused review, AI-assisted for recovery-exit logic; the document states "this remains an internal review, not an external cryptographer sign-off"	two Critical shielded findings closed structurally by the 125,000 sunset; the MatMul amortized-mining finding (High) closed by nonce-bound seeds at 125,000; "independent external assurance still recommended"
2026-07-16	external audit of the MatMul v4 fork (pull request 89), verdict NO-GO, with remediation status	an external reviewer, unnamed in the document	six Critical/High consensus items listed and mapped; activation stayed disabled pending hardware qualification, calibration and review; MatMul v4.7 later activated on 10 August 2026 in the released v0.33.2 node
2026-07-19	independent hardening audit of MatMul v4.4-LT	independent adversarial review, unnamed	"PUBLIC ACTIVATION NO-GO" at the time; byte-exact Metal results recorded; blockers enumerated
ongoing	formal verification of shielded value soundness	`formal-verification/` in the repository	21 machine-checked obligations across an accounting firewall, verifier-relation binding and a reduction to Module-SIS

A project that publishes its own NO-GO verdicts, names the consensus weaknesses in its fork before activation, and keeps the tracker after closing them behaves the way you want consensus developers to behave. What the table does **not** contain is a named cryptography firm's signature on the P2MR, ML-DSA and SLH-DSA stack. The project says so plainly. So do we.

4.6 Our own audits (the layer we can vouch for personally)

We commissioned adversarial, execution-driven audits of the three things we ship. The full reports live in our repositories, which are not yet public; this article quotes them, and they will be published with the open-source release of the security-critical core together with the independent human audit it still needs.

PQ Wallet client, 25 July 2026 (full source, Rust backend plus frontend, target v0.30.0). Verdict: **no exploitable vulnerability found**. The auditor called it "one of the most carefully hardened client codebases I have reviewed" and noted that the residuals the developer cannot close are

documented in-tree rather than papered over. Controls probed and held: triple-gated network egress with no redirects and two-layer response caps; zero DOM injection sinks, enforced by a CI test; whole-frontend SHA-256 pinning with a completeness test so no unpinned script can ship; Argon2id sealing (64 MiB, 3 passes) with random salt and IV; seal-verify-before-overwrite so a wallet can never be bricked by its own upgrade; unlock throttling; a qID login proof structurally unable to authorize a spend. Ranked findings: F1 (High, architectural) the wallet trusts its chain-data node for "you were paid" because it has no SPV; F2 (Medium, by design) an unencrypted wallet option stores the seed in plaintext; F3 to F6 lower. 320 tests green.

Backend infrastructure, 25 July 2026 (btxscan.io and api.btxscan.io, qid.dev, the qID Connect server core, the dashboard, pq-wallet.com). Verdict: **no exploitable vulnerability across the backend**. The qID Connect authentication core held under attack: pinned reference verifier used verbatim; HS256 session tokens with the MAC checked before claims are parsed; nonce provenance checked before any post-quantum math, so unknown-nonce spam cannot burn CPU; atomic single-use consume; login-CSRF guards. The relic claim and mint path held: address-bound challenges, one claim per address and per identity, race-safe item assignment, the issuer key never on the public server. Findings: I1 (carryover of F1), I2 a DNS-rebinding time-of-check/time-of-use gap in an outbound probe, I3 to I5 low.

Hardening applied, same day. Four fixes shipped and tested, 199 tests green across four repositories: the DNS-rebinding gap closed by pinning the validated IP to the dialed connection; a strict Content Security Policy and security headers on play.qid.dev; a production footgun guard in the SDK; a report-only CSP staged on btxscan.io.

qID identity core, six internal rounds, 30 May to 17 August 2026. An independent cryptographer review on 26 May found no functional bugs and confirmed the construction: KEM-DEM encryption, key derivation, authenticated encryption, domain separation across login, attestation and spend, the hash-based cold root versus lattice hot key split, and the byte-exact BTX integration. Later rounds, each executing code against hostile inputs rather than reading it, found and closed lifecycle bugs. The one that matters most: a **QR sign-in takeover** in qID Connect (a bystander who could see the screen could sign the victim into the attacker's account) was found about a month after launch, reproduced over real HTTP, fixed by changing the client contract, and published in the changelog together with a correction of a prior public claim. The maturity assessment, written by reviewers who did not do the work and including one briefed to argue the score was too high, scores qID at **6 of 10, ceiling 7 without a third-party audit**, with "independent assurance" pinned at 1 of 10.

The money path against a real node, 17 August 2026. All five regtest suites ran against the official `btxd` v0.33.2 binary (checksum-verified):

```
harness      testmempoolaccept {"allowed":true,"vsize":3873}  scriptPubKey byte-exact
lifecycle    built -> broadcast -> mined 1 conf, dest received 1.5 BTX
multiinput   2-input accepted (vsize 7691); tampered amount REJECTED
             ("Invalid ML-DSA signature") - the negative control holds
select       fee == vsize*rate, value conserved, clamp works
```

```
sweep        single output = total − fee, exact
vsize        estimate EXACT for an unseen 3-input spend (11509 vB)
```

The statement the qID team published to survive a hostile reader is the right way to end this part:

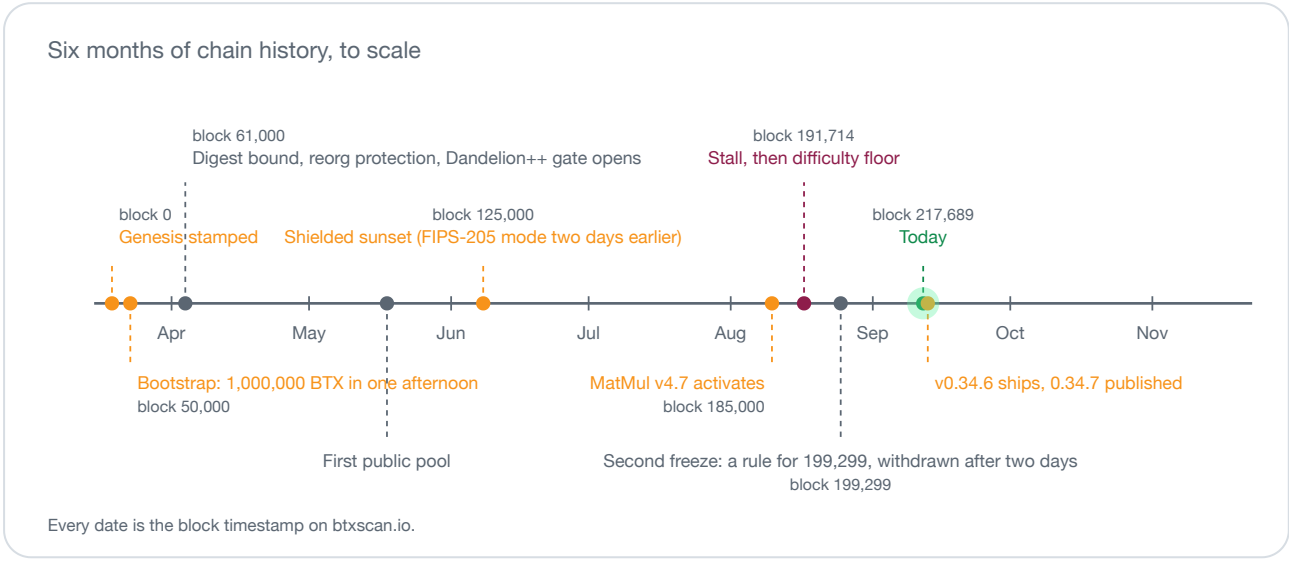
> qID is a post-quantum identity construction that has been through six internal hardening rounds, each conducted by executing the code against adversarial inputs rather than reading it. It has 100 tests including NIST ACVP vectors for every primitive, a CI gate proving the shipped bundle rebuilds byte-identically from reviewable source, and canonical binary signing formats intended for independent reimplementation. Every finding from those rounds is recorded, including the ones that were refuted and the fixes that were wrong before they were right. It has never been audited by anyone outside the project. It is deployed only in gated, low-value contexts. Anyone evaluating it for real value should treat the internal review as evidence of diligence, not as assurance, and should commission an independent audit before that changes.

An audit brief for engaging an external firm on qID Connect exists, with a fixed scope of about 6,000 lines and seven claims we pay the auditor to attack. It has not yet been commissioned.

Part 5. Network status and evolution

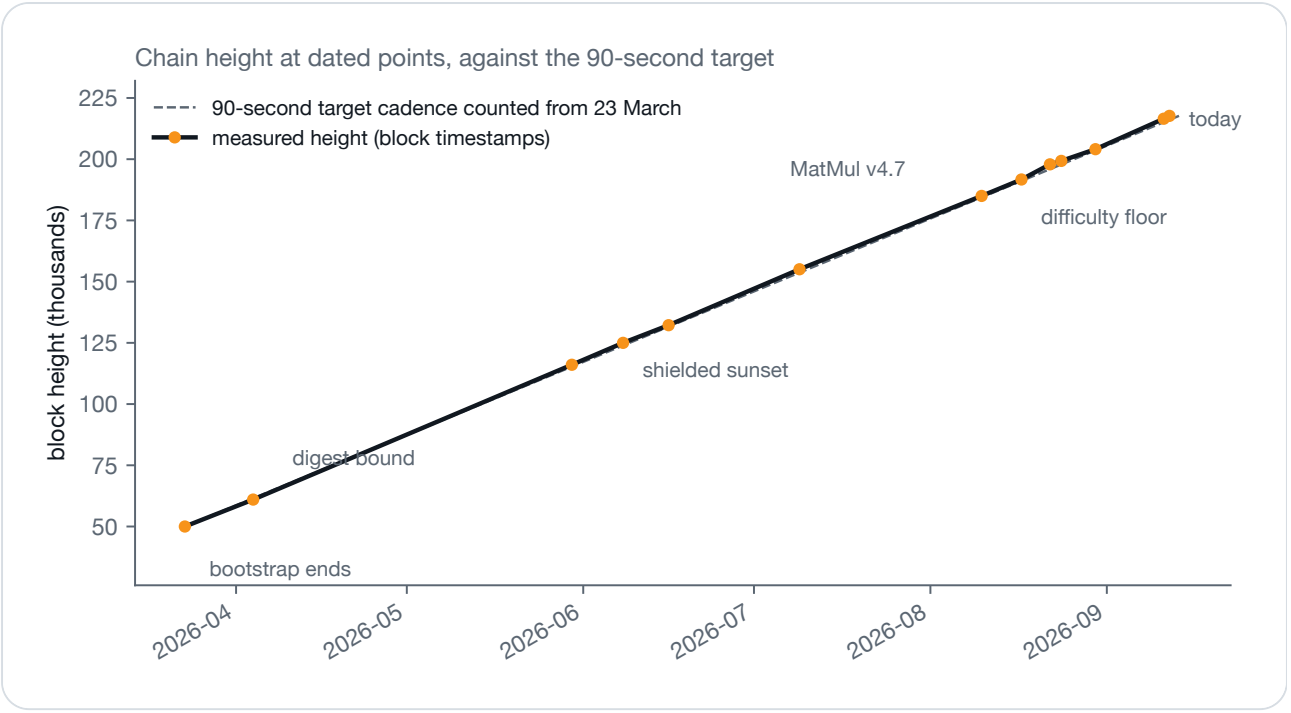
Everything in this part is measured: from BTX's own consensus source for heights and rules, from btxscan.io for dates, from the public node census and pool endpoints that easybtx.com/stats reads, and from the weekly history series at easybtx.com/api/network-history. Refresh any figure before you reuse it; the live page is the source, this is the snapshot.

5.1 The chain's history, block by block



Six months of chain history to scale, from the genesis stamp on 19 March 2026 to the present. Every date is the block timestamp read from btxscan.io rather than a third-party feed. Earlier revisions of this figure placed Dandelion++ in the future at block 250,000, which is what the project's README says in two places. The source disagrees: the relay gate is tied to the shielded hardening height and has been open since block 61,000.

Date	Height	What happened
19 Mar 2026	0	Genesis stamped. Post-quantum signatures and MatMul proof-of-work apply from block zero.
23 Mar 2026	1	First block actually mined, four days after the genesis stamp.
23 Mar 2026	50,000	Bootstrap window ends: 1,000,000 BTX in about six and a half hours at a 250-millisecond cadence (Part 12, item 3). ASERT takes over. ASERT takes over.
4 Apr 2026	61,000	The MatMul product digest is bound into the block; shielded hardening fork; reorganization protection begins; the Dandelion++ relay gate opens.
18 May 2026		First public mining pool goes live.
24 May 2026		easyBTX v1, the one-click miner, released.
6 Jun 2026	123,000	C-002: FIPS-205 pure-mode SLH-DSA verification becomes consensus.
8 Jun 2026	125,000	The shielded sunset. Consensus stops accepting new shielded deposits; MatMul seeds become nonce-bound.
14 Jun 2026	130,500	Seeds bound to the parent block, closing a template-replay shortcut.
10 Aug 2026	185,000	MatMul v4.7 activates. One attempt becomes a 30-second episode. Mac mining pauses for nine days.
17 Aug 2026	191,690	The signed frontier stalls for hours. v0.33.3 ships the same day.
17 Aug 2026	191,714	A difficulty floor becomes consensus. Nodes below v0.33.3 park at 191,713.
24 Aug 2026	199,300	The height at which the v0.34 releases later close the shielded pool in both directions; whatever remained inside is treated as burned.
25 to 27 Aug 2026	199,299	A change to a difficulty rule, published on 25 August for block 199,299, parks the nodes that take it just below that height while the rest of the network mines on; the reference repository records blocks accepted up to 199,328 while both operator nodes sat at 199,300. Upstream withdraws the rule two days later.
27 to 30 Aug 2026		v0.34.1 to v0.34.5, the "handover" reference cuts: the withdrawn rule removed, the 199,300 pool closure shipped, the assumeutxo pin moved to 201,500.
12 Sep 2026	217,689	Today. 4,353,780 BTX mined, 20.7 percent of the cap.



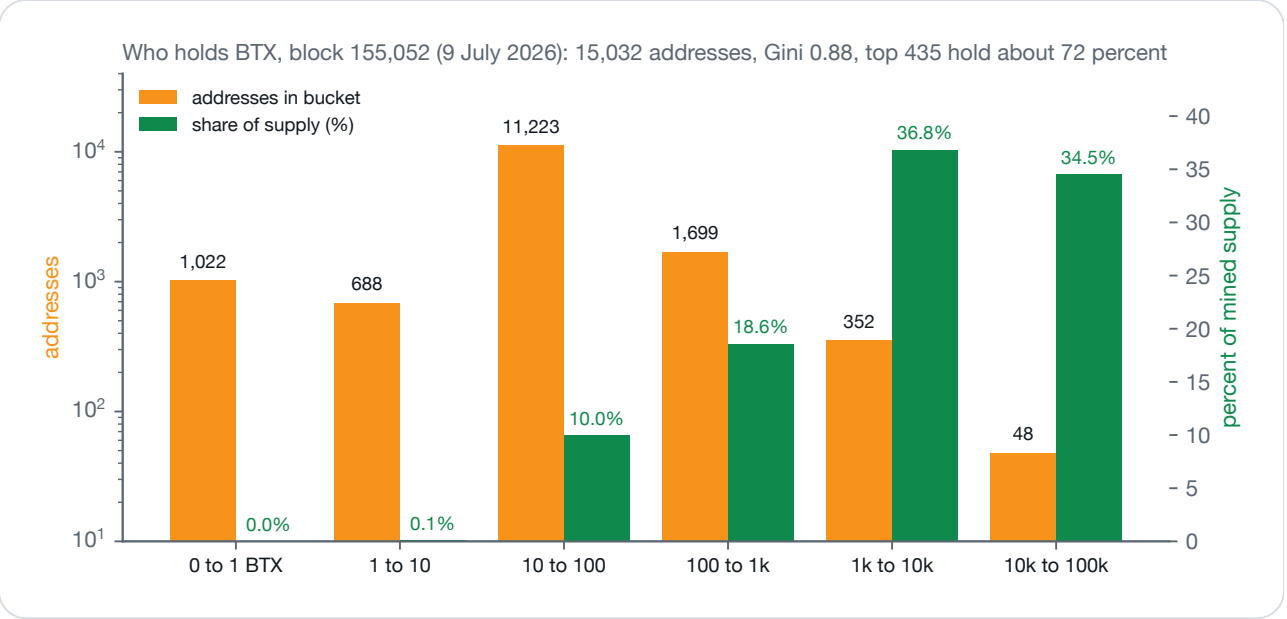
Chain height at dated points, against the 90-second target cadence counted from the first mined block. Every point is a block timestamp read from btxscan.io. The V1 chart carried a 30 August point taken from btxprice.com, whose own node had parked at 199,294; the chain was at 204,088 that day, and the point is corrected here.

The chart shows the one distortion worth understanding: the bootstrap put the chain 50,000 blocks ahead on day one, so the lifetime average block time is about 69 seconds, while the interval since the bootstrap ended is about 89 seconds, right at target. The weekly history series measured a 7-day average of 92.9 seconds on 11 September 2026.

5.2 Growth, measured

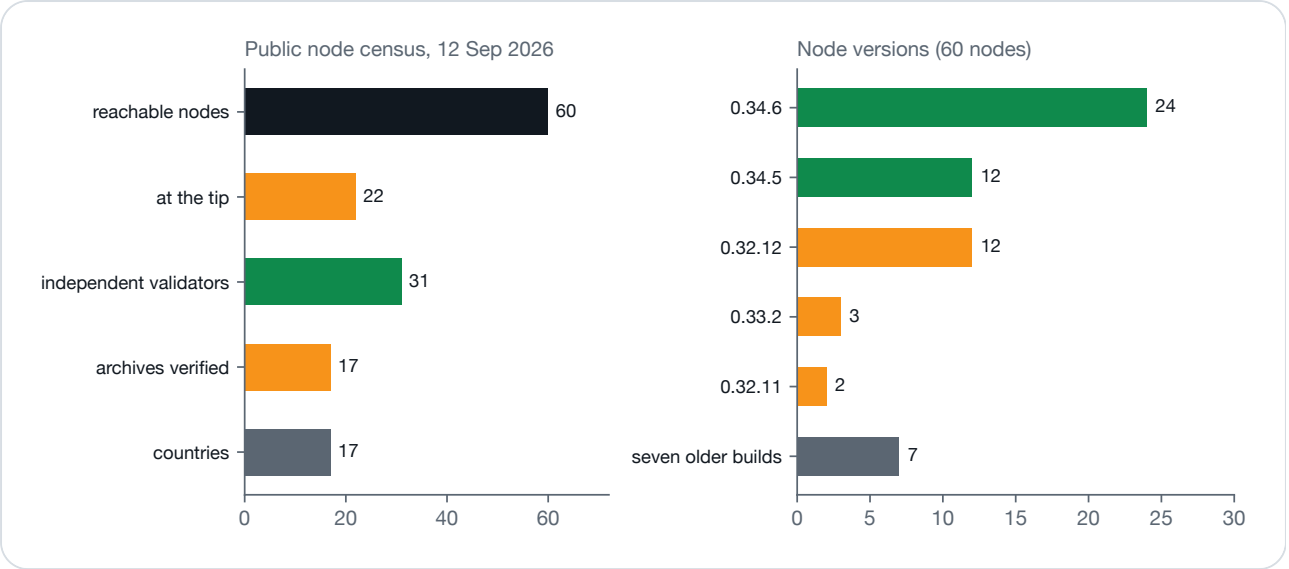
Date	Height	Coins mined	Nodes seen	Other measurements
30 May 2026	~116,050	~2.32M		about 41 active miners; nearly every block coinbase-only (our measurement against the then-public explorer)
16 Jun 2026	132,209	2.64M		3,092 addresses holding a balance
9 Jul 2026	155,052	3.10M		15,032 holding addresses; 62,602 unspent outputs; shielded pool down to ~21,000 BTX (0.7 percent)
18 Aug 2026	~192,000	~3.84M	about 60 (54 discovered plus 8 curated)	more than 10 countries; one GPU attestor signing the frontier
22 Aug 2026	197,897	3.96M	65 live, 16 countries, 30 at tip, 12 archives verified	two working pools with 53 and 670 connected workers
11 Sep 2026	216,528	4.33M	62 total, 25 at tip, 32 independent, 17 countries	7-day average block interval 92.9 s; one pool alive
12 Sep 2026	217,689	4.35M	60 total, 22 at tip, 31 independent, 17 countries, 17 archives verified	one reachable pool (BTX Pool, 1.5 percent fee, 7 workers); Byron Bay unreachable at the time of the check

Two readings. First, coins mined track height exactly at 20 BTX per block, because no halving has happened and there is no other issuance path. Second, the holder count jumped almost fivefold in the 24 days to 9 July, mostly because the shielded pool unwound into thousands of new transparent addresses of 25 to 30 BTX each; an address is not a person, so read that as on-chain activity, not a headcount. The July census also measured concentration: the largest 435 addresses held about 72 percent of mined coins and the Gini coefficient was 0.88, which is what a fairly mined coin looks like at four months old (Bitcoin was more concentrated at the same age), and many of the largest addresses are pool and exchange wallets.



Who holds BTX, measured from a full node's unspent-output set at block 155,052 on 9 July 2026: 15,032 addresses, with the count per size bucket (orange, logarithmic) and each bucket's share of the mined supply (green). Most addresses hold 10 to 100 BTX, a signature of the shielded pool unwinding; a few dozen addresses hold thousands each. An address is not a person; many of the largest are pool and exchange wallets. Source: the easyBTX holder census.

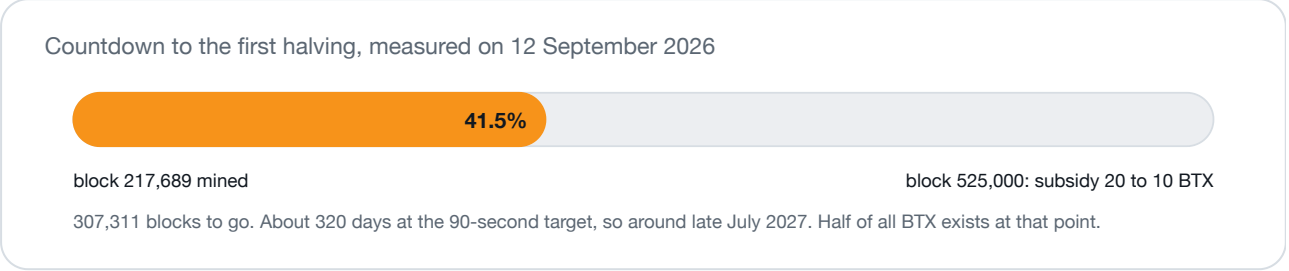
Node versions in the 12 September census: 0.34.6 on 24 nodes, 0.34.5 on 12, 0.32.12 on 12, and twelve nodes spread across older builds. When that census was taken, the newest tagged release was v0.34.5 and the 0.34.6 builds in the wild came from untagged later cuts. v0.34.6 was tagged the next morning; Part 9 begins there.



The public node census on 12 September 2026, from easybtx.com/api/network: 60 reachable nodes in 17 countries, 22 at the tip, 31 validating the proof-of-work independently, 17 archives verified by a real attestation request. Right: which node software those 60 nodes run.

Two data notes. The only third-party public history feed for BTX, btxprice.com, stopped updating on 30 August 2026, and its own node had parked at height 199,294 in the freeze below 199,299, so its last figures describe a stalled node rather than the chain. That is why easyBTX started its own weekly series on 11 September. And the two pools measure work in different units (MatMul hashes per second versus inference episodes per second), so no honest "network hashrate" figure can be summed across them; this dossier does not quote one.

5.3 The halving schedule

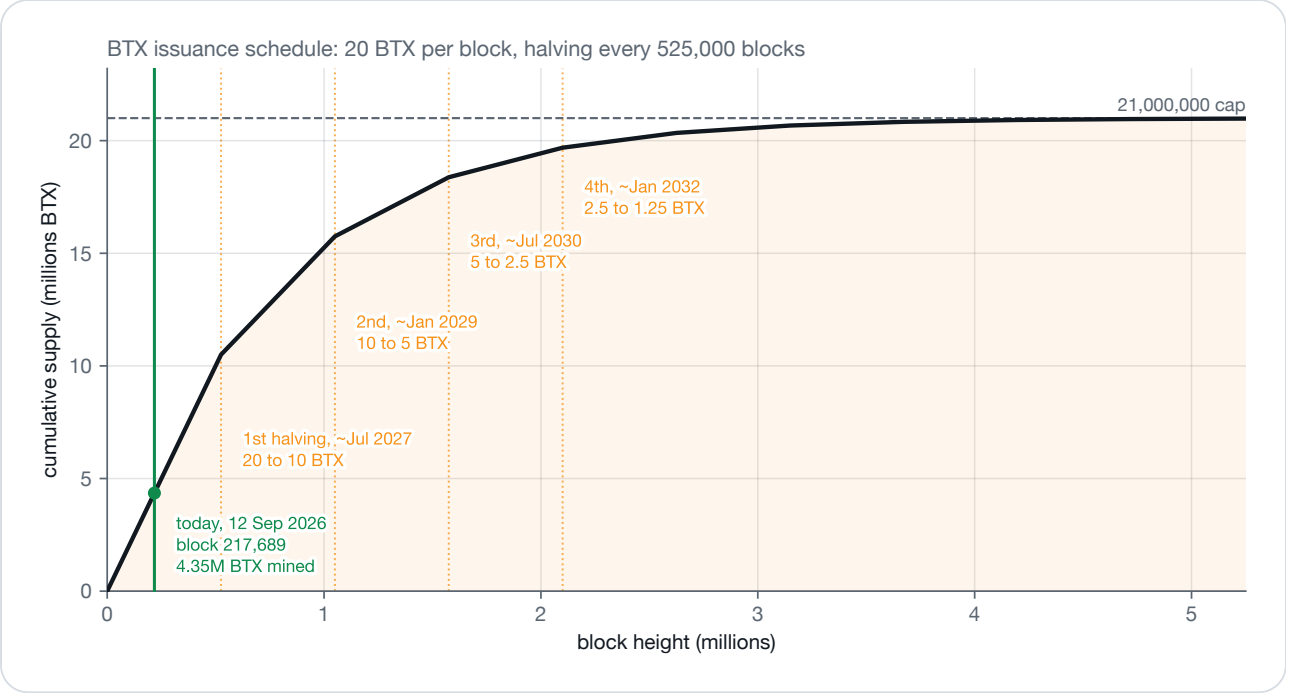


Faster blocks pull the date earlier and a stall pushes it later, so treat the date as a target rather than a schedule. On the web page the bar fills as it scrolls into view; in print it is static.

BTX has not had a halving yet. The subsidy is 20 BTX, and 307,311 blocks remain to the first halving at block 525,000. At the 90-second target that is about 320 days, so **around late July 2027**. Blocks since the bootstrap have arrived close to target, so that estimate is reasonable but not a promise; a period of faster blocks pulls it earlier, a stall pushes it later.

Era	Blocks	Subsidy	Coins issued in era	Cumulative supply	Era starts (estimate at 90 s)
1	0 to 524,999	20 BTX	10,500,000	10,500,000	19 Mar 2026 (actual)
2	525,000 to 1,049,999	10 BTX	5,250,000	15,750,000	~29 Jul 2027
3	1,050,000 to 1,574,999	5 BTX	2,625,000	18,375,000	~Jan 2029

Era	Blocks	Subsidy	Coins issued in era	Cumulative supply	Era starts (estimate at 90 s)
4	1,575,000 to 2,099,999	2.5 BTX	1,312,500	19,687,500	~Jul 2030
5	2,100,000 to 2,624,999	1.25 BTX	656,250	20,343,750	~Jan 2032
6	2,625,000 to 3,149,999	0.625 BTX	328,125	20,671,875	~Jul 2033
7	3,150,000 to 3,674,999	0.3125 BTX	164,062.5	20,835,937.5	~Jan 2035
8	3,675,000 to 4,199,999	0.15625 BTX	82,031.25	20,917,968.75	~Jul 2036
...	halving every 525,000 blocks	halves each era		converges on 20,999,999.93	roughly every 1.5 years



The issuance schedule: cumulative supply against block height, with the first four halvings marked and today's position. Dates are estimates at the 90-second target from block 217,689 on 12 September 2026. Source: the consensus constants in src/kernel/chainparams.cpp.

For comparison, Bitcoin halves every 210,000 blocks, roughly four years. BTX's 525,000-block interval at a 90-second target is roughly 1.5 years, so the emission curve compresses Bitcoin's 130-year schedule into about 50 years. Half of all BTX will exist after the first halving, in mid-2027. Three quarters after the second, in early 2029.

5.4 Where to watch it live

- easybtx.com/stats: chain height, issuance against the cap, halving countdown, block-time averages, node census by class and version, every working pool with its own units, refreshed every 60 seconds from easybtx.com/api/network (JSON, no key).
- easybtx.com/api/network-history: the weekly series, one measured row per week since 11 September 2026: height, issuance, tip difficulty, the real 7-day block interval, node census and per-pool figures. Dead upstreams are recorded as null, never fabricated.
- easybtx.com/nodes: a live map of every reachable public node.

- btxscan.io: the explorer, for any block, address or transaction.
- The week the chain froze, and the day it came back: the measured account of the August stall and recovery, block by block.

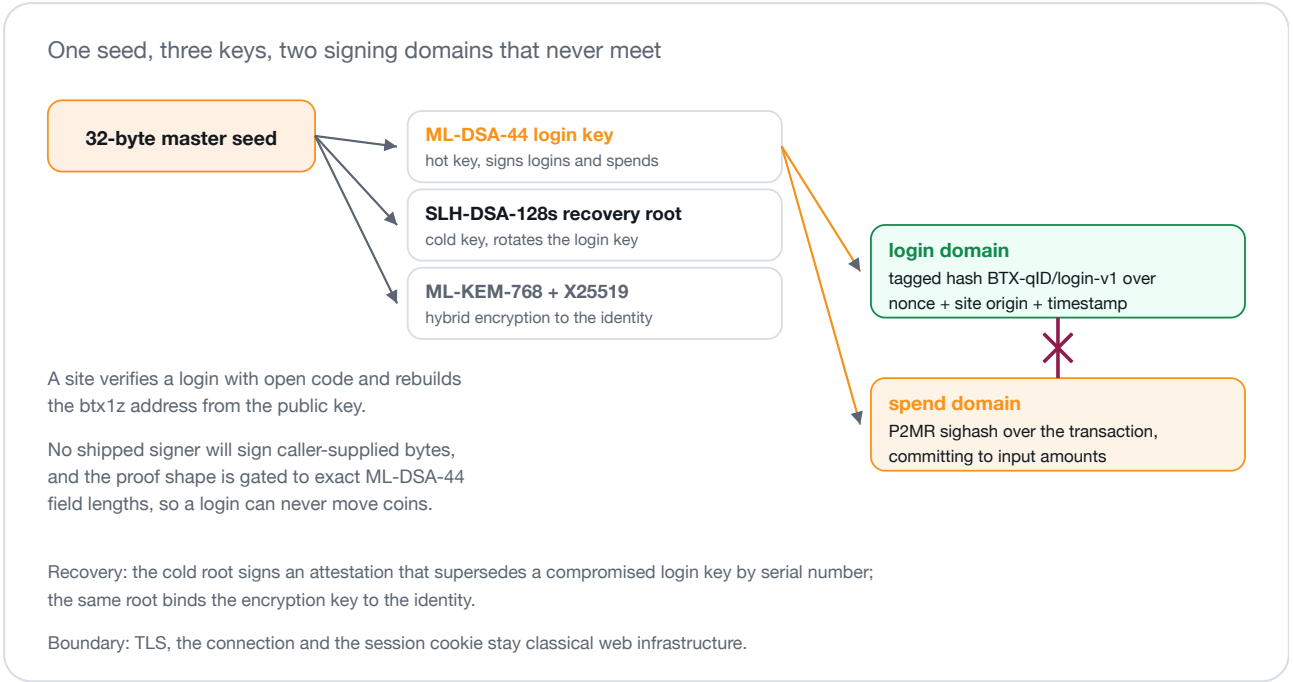
Part 6. qID: post-quantum identity, and what it cannot do

Bitcoin never got a native "sign in with your key" that ordinary websites use. BTX has one, and it is built from the same keys as the chain.

What it is. A user's qID keys are the same ML-DSA-44 login key and SLH-DSA cold recovery root that control their BTX coins, in the same encoding: "byte-exact to the chain". Signing in to a site means producing an ML-DSA signature over a one-time, origin-bound challenge; the site verifies it with open-source code and rebuilds the `btx1z` address from the public key. No password, no shared secret stored by the server, and no relay server in the middle: the site's own backend issues the challenge and receives the proof.

Why a login can never move coins. Login signatures are made over a tagged hash in the `BTX-qID/login-v1` domain. Transactions sign under the P2MR sighash. There is no path in any shipped signer to sign caller-supplied bytes. The proof shape is structurally gated (exact ML-DSA-44 field lengths), so a raw seed or an arbitrary blob is refused. This was one of the properties our own July audit set out to break; it held.

Why it cannot be done with a classical key. The verifier accepts only an ML-DSA-44 signature and derives the address from the post-quantum public key. BTX has no classical address type to build one from. That property is unusual and holds up to inspection.



One seed, three keys, two signing domains. qID derives the ML-DSA-44 login key, the SLH-DSA-128s recovery root and a hybrid ML-KEM-768 plus X25519 encryption key from one 32-byte seed. A login signs a tagged hash in the BTX-qID/login-v1 domain over the nonce, the site origin and a timestamp; a transaction signs the P2MR sighash. Neither verifier accepts the other's digest, so a login proof can never move coins. Source: the qID security documentation.

The rest of the construction. An SLH-DSA cold root that can rotate a compromised login key through a recovery-signed attestation with expiry and serial numbers; hybrid **ML-KEM-768 plus X25519** encryption to an identity, with the KEM key bound to the identity by the cold root (public-key substitution was tested and refused); passkey sealing through WebAuthn with the PRF extension, user verification required, origin-bound, and failing closed if the platform lacks PRF; a 90-day coordinated disclosure policy with response targets.

The honest boundary, in the project's own words. A post-quantum login makes exactly one thing post-quantum: the proof that a user controls that address. The TLS certificate naming the site, the connection, and the session cookie afterwards are ordinary web infrastructure, because browsers have no post-quantum certificate authority yet. And a site you sign into learns your BTX address, which is a public ledger entry. That is a real trade-off, and it is not anonymity.

Status. qID Connect SDK 1.0 shipped in July 2026; the server package now at 1.7.0 runs on btc2btx.com, on easybtx.com's gated node directory, and on btxscan.io's relic claims. The release pack is MIT and downloadable from build.qid.dev; the source repositories are not yet public. Every live integration today belongs to the same estate as this article; there is no independent third party in production yet.

Part 7. PQ Wallet: holding post-quantum coins on a desktop

PQ Wallet is the self-custodial desktop wallet (macOS, Windows, Linux) for BTX; the bonuz wallet covers iPhone and Android. Current version **v1.3.0 (6 September 2026)**. Architecture in one paragraph: a Rust process makes every network call through a single allow-listed proxy; the webview that holds the keys never touches the network, has no HTML injection sink (enforced in CI), ships under a strict Content Security Policy, and every shipped script is SHA-256 pinned. Seeds are sealed with Argon2id plus AES-GCM behind a passphrase or Touch ID. The send core is the node-verified qID transaction builder from section 4.6.

What the July audit said the wallet cannot do to you. A compromised chain-data server cannot steal funds, because the P2MR sighash commits to input amounts: lie about a UTXO's value and the network rejects the transaction. Fees are hard-clamped. Recovery files are written with owner-only permissions. A new seal is proven to reopen before the only copy is overwritten.

What it can do to you, and is documented. Without SPV (simplified payment verification against block headers), a compromised chain-data node can show a fabricated confirmed *incoming* payment. It cannot take anything; it can lie about you being paid. Mitigations shipped since: an independent second source checks for chain forks and says so when it cannot run; every send is broadcast to every reachable node at once (measured on 6 September: two public nodes' mempools shared nothing, so a single-node broadcast was a bet); read-only sources are scoped to exactly the two block requests they need. The full SPV fix is on the list, not done.

What is deliberately not there. No in-app auto-updater, by decision: an updater signing key is a mass remote-code-execution risk for a money wallet, and manual download with a published SHA-256 suits self-custody. Shielded transactions are not planned in this wallet. Code signing and

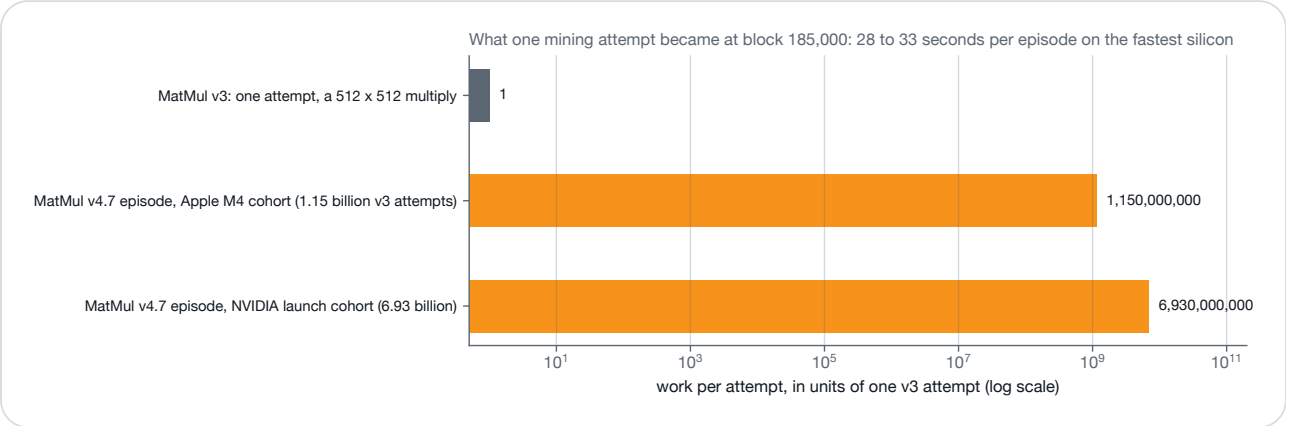
notarization of the builds are still pending, which is why some antivirus engines flag unsigned builds and why every release publishes a checksum.

Part 8. MatMul proof-of-work, and the AI angle, stated carefully

8.1 What miners compute

MatMul v3 (genesis to block 184,999). One attempt was a 512 by 512 matrix multiplication over the Mersenne prime field F(2^31 − 1), chosen because that field maps cleanly onto 32-bit integer arithmetic on both CUDA and Apple Metal. The matrices expand deterministically from seeds in the block header; low-rank noise derived from the header and nonce prevents precomputation; a field-algebraic inner-product compression shrinks the 33.5 MB result to about 131 KB, which is then SHA-256d hashed against the target. Verification used Freivalds' algorithm (two random rounds, false-positive probability below 2^−62), so checking a block cost a few hundred bytes of arithmetic. Overhead above a bare multiply was about 16.5 percent. The construction follows the 2025 paper "Proofs of Useful Work from Arbitrary Matrix Multiplication" (Komargodski, Schen and Weinstein, arXiv:2504.09971).

MatMul v4.7 Profile 1 (from block 185,000, 10 August 2026). One attempt, now called an episode, runs four sequential rounds of a sixteen-layer feed-forward network at consensus dimension 4,096 with batch-sequence 16,384: **141,149,805,215,744 exact multiply-accumulates** in integer arithmetic, holding about 4.8 GB resident on the accelerator. Measured over 100 production-shaped headers with zero CPU fallback: 28.2 seconds at the 99th percentile on an Apple M4 Max (Metal) and 32.7 seconds on a Blackwell-class 16 GiB NVIDIA card (CUDA). The release-sealed CUDA and Metal cohorts produced byte-identical headers and digests. The block header stayed a fixed 182-byte digest-only object, which is why pools and wire protocols survived the fork intact.



What one mining attempt became at block 185,000, on a logarithmic scale. BTX's own fork calibration measured one v4.7 episode as the work of 1.15 billion v3 attempts on its Apple M4 cohort and 6.93 billion on its NVIDIA launch cohort; an episode takes 28.2 seconds at the 99th percentile on an M4 Max and 32.7 seconds on a Blackwell-class 16 GiB card. Sources: the MatMul v4.7 specification and the BTX Handbook.

8.2 Why a chain would do that

- **Indivisible attempts.** An episode cannot be half-done or guessed cheaply. To attempt a block you must commit a modern accelerator to a modern computation for tens of seconds. The work

stopped resembling a lottery.

- **ASIC misalignment, stated precisely.** The specification does not claim ASICs are impossible. It says they are economically misaligned with commodity AI and GPU hardware, because the workload is identical to the dense integer GEMM (general matrix multiply) that tensor cores are already optimized for, leaving little headroom for a special chip.
- **Rentable-GPU attacks get more expensive.** The design documents name pricing the cheap hardware tail out of rental viability as an explicit goal. Renting consumer GPUs is the standard way small proof-of-work chains get 51-percent attacked (Bitcoin Gold twice, Vertcoin three times).
- **Admission by measurement, not by list.** A device qualifies to mine by returning a byte-exact answer to a computation the software already knows, in under a second. There is no chip allowlist. An earlier rule requiring a particular Apple chip was wrong and was removed after measurement.
- **Security hardware that remains productive.** Bitcoin's SHA-256 summoned a fleet of single-purpose ASICs. BTX's work summons general-purpose accelerators that can leave mining and run open models, agents or numerical workloads. The protocol paper frames this as capital that is not stranded when mining conditions change.

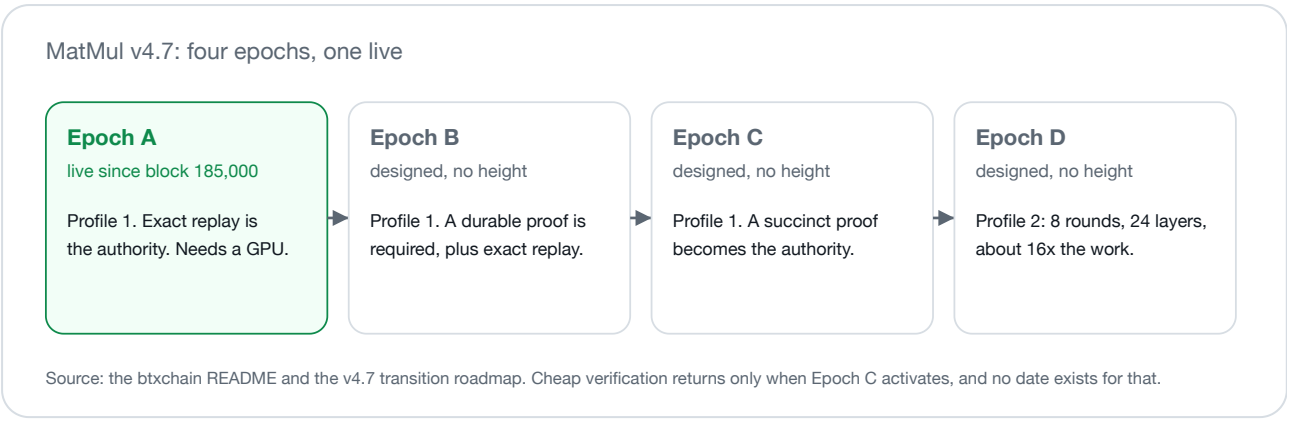
8.3 The institutional thesis, in the project's words

"Own AI for the expansion. Own BTX for the contraction." The claim is not an inverse beta to AI markets. It is that an institution can separately govern a scarce BTX reserve, compatible accelerator capacity that has an alternative BTX workload under stress, and contractual switching and step-in rights. Holding BTX alone creates no claim on infrastructure or mining revenue; the paper says so, and it lists the risks of each channel (basis, liquidity, device obsolescence, power, custody, regulatory, concentration). The same page carries a section titled "claims to avoid", including that post-quantum design does not eliminate implementation, custody, governance or market risk.

8.4 What is true today and what is not

Claim you will hear	Status
"BTX mining is AI inference"	The workload is transformer-shaped exact-integer arithmetic. It is not inference of a real model on anyone's data. Inputs are seeded from the chain; only a digest is kept.
"BTX sells AI compute"	No, and the published 0.34.7 code strengthens the answer rather than weakening it. Its release notes remove remote and paid inference from the roadmap, its README states that BTX does not sell remote inference and exposes no inference endpoint, and a test asserts the remote-inference capability reports false. Inference is local after you have the file. What that proposes to price is the delivery and the release of model files, never inference. A verifiable-compute market remains a later direction named in the specification. See Part 9.
"The hardware is the same as AI hardware"	Yes. Qualified Metal and CUDA accelerators, the same tensor-core class that runs inference.
"BTX has a shipping AI-facing product"	Yes, one: the MatMul **service-challenge** RPCs, an admission-control gate the documentation describes as an AI-agent CAPTCHA. A gateway can require a fresh, chain-bound work proof before an expensive route runs. It gates AI; it does not run it.
"Verification is cheap"	Under v3 it was (Freivalds). Under v4.7 Epoch A it is a full exact replay, so it needs a qualified GPU. Succinct proofs (Epochs B to D) are designed and have no activation height.

Claim you will hear	Status
"The proof-of-work is post-quantum"	Not a claim the project makes. MatMul's hardness rests on a direct-product conjecture for random matrix products, not on a post-quantum assumption. The relevant point is the one from Part 1: Grover barely touches proof-of-work; Shor is what the signatures had to survive, and they do.



What the sequence costs a node operator, which the boxes do not say. Until Epoch C, the only way to check a block yourself is to replay the work on a qualifying GPU. Every node without one is following somebody else's verdict, and no height exists for the epoch that would end that.

8.5 What the fork cost

Mac mining stopped for nine days until a Metal solver existed. Nodes that had not upgraded stopped at 184,999. Cards older than the engine vendor's floor could no longer mine. Displayed difficulty dropped by a factor of about 120,000 at activation and took some 40 blocks to settle. On 17 August the chain stalled at 191,690 and recovered the same day through v0.33.3's difficulty floor at 191,714. Every one of those events is in the release notes and in the research archive, dated.

The second question: where BTX is going, and who decides

Everything up to here answers the question this dossier was commissioned for. The two parts that follow answer the one it grew into: what the 0.34.7 model network actually contains, now that it is a public pull request rather than an announcement, and who can ship anything at all. Part 11 onward judges both halves together.

Part 9. The model network: from an announcement to a pull request in one day

On 13 September 2026 this chain's lead developer published a release and, a little later the same day, described a much larger one that existed only on his own machine. On 14 September he published that too, as a pull request anyone can read. This part was written in the gap between those two days, so the predictions it made from the announcement can now be checked against the code. Where they were right this part says so, and where this document was wrong it says that first.

9.1 What happened, on both days

Three timestamps, all checkable.

At **08:03 UTC** commit `3013c2c2` was authored. At **08:31 UTC** it merged to `main` as `b91ed32f`, closing pull request 135. At **09:18 UTC** it was published as **v0.34.6**. The release notes lead with a line that read, that morning, like ordinary protocol hygiene: "SHA-256 is the only new HTLC. Use `htlc_sha256(<32-byte digest>, <claimer>)`."

Later the same day, in the project's Telegram group, the same developer posted a long description of a release numbered **0.34.7**, which he called a "Native Model Network", together with an implementation report and the sentence "Expect 0.34.7 to be launched in the next 24 hours."

0.34.7 is now pull request 156, opened against `main` at 11:57 UTC on 14 September with four commits: 240 files, 33,367 added lines, 12 deleted. Those are the figures as it opened, and they are already out of date. Three more commits landed within thirty hours, so any size quoted here is a snapshot. The fourth command in section 9.8 returns today's. It is open, unmerged and untagged, and the branch sets `CLIENT_VERSION_IS_RELEASE` to false, so the newest release is still v0.34.6. A search of the public tree now finds all three of the terms that previously returned nothing, and `btx-modeld` is a real build target.

One detail deserves to be singled out, because it is the strongest thing this document can say about the developer's honesty. The implementation report of 13 September gave its private HEAD as `b094c6ba420f...`, and this article published a command showing that GitHub returned "No commit found" for it. That commit is now the third of the five commits in the pull request, authored on the date the report claimed. The report named a commit that nobody could see, and when the code appeared, it was that commit. Whatever else is uncertain here, the private report was describing something real.

A note on sourcing, needed here more than anywhere else in this document. The design described in this part, as opposed to the code that implements it, comes from messages in a group chat, which is not an archive: messages can be edited or deleted and a reader in six months may not retrieve them. Everything about v0.34.6 comes from the public git tree, which anyone can check at any time. Where the two disagree, believe the tree. We report the announcement because the community is already discussing it, and a document that claims to be an honest reference should not pretend the largest claim of the week was never made. We report it as a claim.

Every part of the implementation report that can be checked against the public tree checks out. The baseline commits are right, the pull request number is right, and the regression count it quotes for "HTLC templates plus PQ policy plus netbase, 66 of 66" reproduces exactly at that revision. The report is accurate wherever accuracy is verifiable. That is worth saying, and it is also the reason to be careful: accuracy in the checkable half is what tempts a reader to accept the unverifiable half.

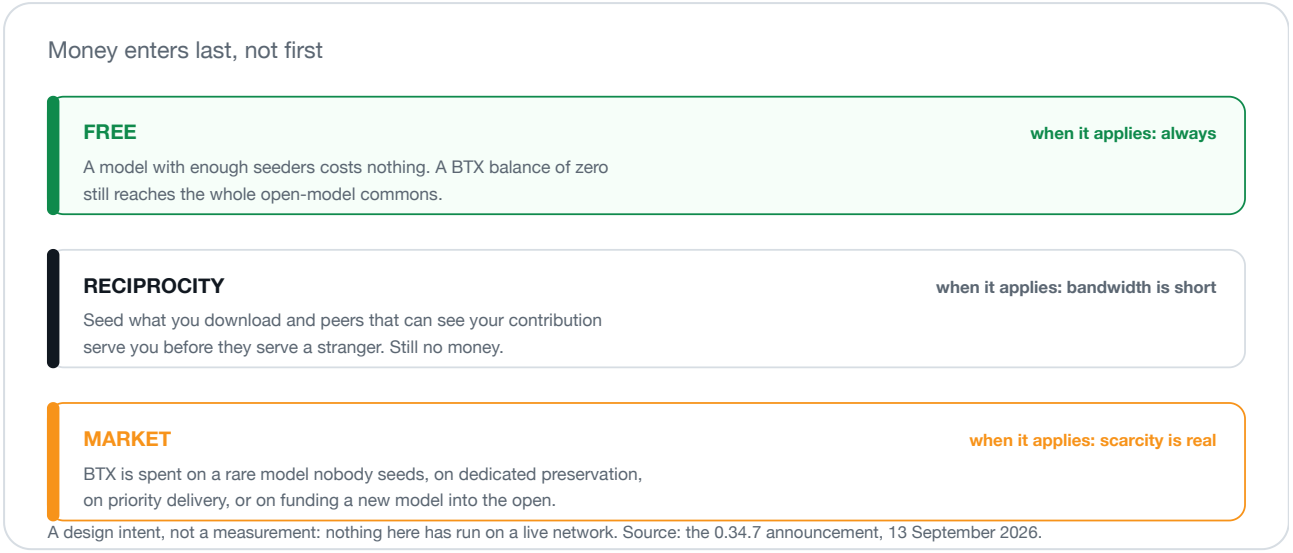
9.2 What the announcement claims BTX becomes

In the developer's framing, BTX stops being only AI-native money and becomes "decentralized infrastructure for open AI". His one-line summary is the clearest sentence in the announcement:

> The node already has the compute. BTX gives it models and money.

Concretely: a model gets an address of the form `btx://...` identifying exact weights by hash. A client resolves it through community relays, fetches pieces from independent nodes, verifies the result locally, stores it, runs it on the user's own accelerator, and optionally keeps seeding it. No central registry decides which models exist and no inference company sits between the user and the model. In his words, "there is no requirement that Hugging Face remain online."

The economy is free by default. Models with enough seeders cost nothing, seeding earns better service from peers under congestion, and money enters only where scarcity is real. He is explicit that this is a refusal to build a toll booth: "a user can participate in the entire open-model ecosystem with a BTX balance of zero."



The proposed economy, as described. Models with enough seeders are free, seeding earns better service from peers under congestion, and BTX is spent only where scarcity is real. The stated aim is that a balance of zero still reaches the whole open-model commons, which is a deliberate refusal to put a toll booth in front of open weights. Nothing here has run on a live network. Source: the 0.34.7 announcement, 13 September 2026.

Two of the premises behind this are worth separating, because they are not equally true. That open-weight distribution is **centralised** is measurable: Hugging Face passed three million public models in August 2026, its terms reserve removal "at any time, at our sole discretion", and when Runway deleted `stable-diffusion-v1-5` in August 2024 it broke code across the ecosystem overnight. That open-weight release is **politically restricted** is much weaker on the current record, and in the United States close to the opposite: the NTIA recommended against restricting open weights in 2024, the relevant export-control entry does not cover published weights, and the 2025 AI Action Plan contains a section encouraging open-weight AI. The genuine restrictions are narrower and specific: the EU's systemic-risk threshold for general-purpose models carries no open-source exemption and the largest Llama models sit above it, China requires pre-launch filing, and Meta's own licence bars EU entities from its multimodal models. That last one is a licence restriction, and a distribution network does not dissolve it. It only makes it easier to ignore.

9.3 The part that shipped, and the name nobody noticed

Here the tree says something the announcement did not.

The financing design rests on a hash-time-locked contract: money claimable only by revealing a secret, refundable if nobody does. That primitive is not a plan. It is in v0.34.6, and confirming it takes about ten seconds. The shipped descriptor is:

```
mr( htlc_sha256(<32-byte SHA-256 digest>, <claimer key>) , refund(<locktime>, <sender key>) )
```

Read that against Part 3 and the shape is familiar. It is the same Pay-to-Merkle-Root output that gives BTX its post-quantum defence by holding an ML-DSA leaf and an SLH-DSA leaf. Here the same tree holds a claim leaf and a refund leaf. One output type, two unrelated jobs, no new consensus rule for either. The claim leaf compiles to `OP_SHA256 <digest> OP_EQUALVERIFY <pubkey> OP_CHECKSIG_MLDSA` and is spent with a two-item witness: a signature and a 32-byte preimage.

The property that makes release financing conceivable is asserted by a functional test shipped with the release, `test/functional/wallet_htlc_atomicswap.py`. Its docstring states the outcome plainly: claiming the output means "the preimage is revealed on-chain in the claim witness". To take the money you must publish the secret. That is ordinarily how atomic swaps work, where the secret unlocks a payment on another chain. The announcement points it somewhere else: let the secret be a decryption key, and the other leg of the swap is not a payment at all. It is publication.

Two details deserve more attention than they have had.

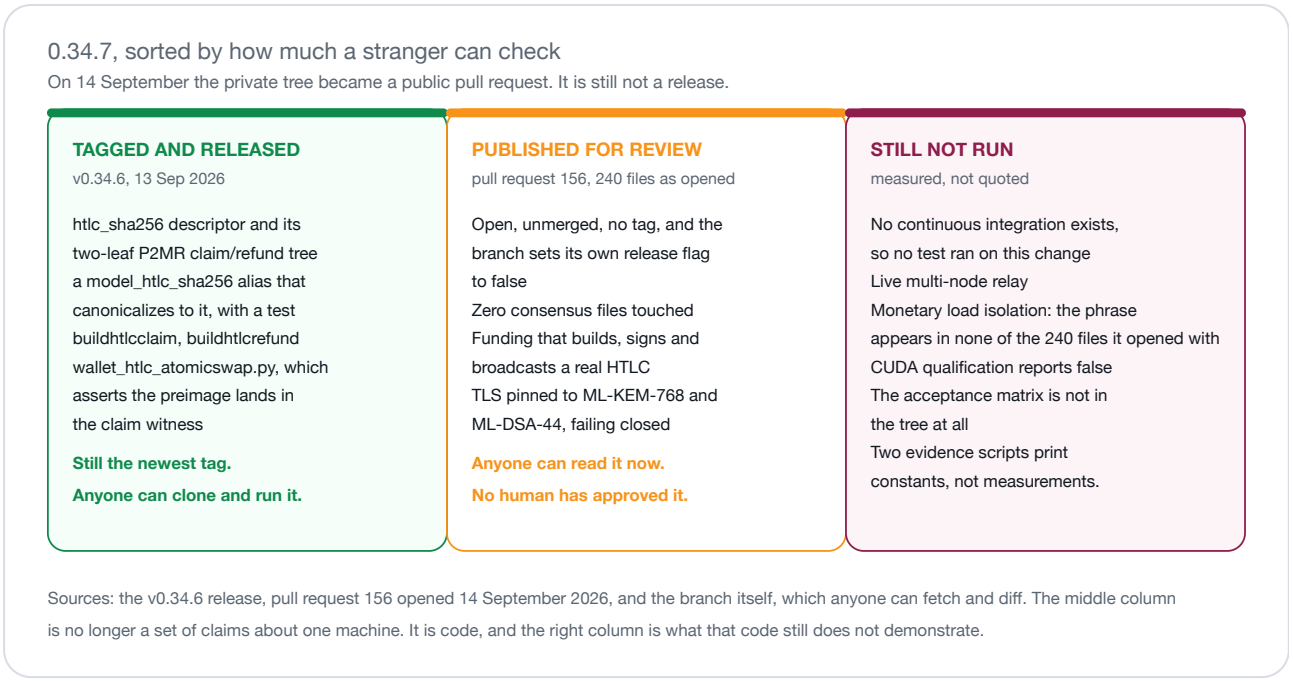
The first is why the lock moved to SHA-256 at all. The header comment in the shipped source gives the reason, and it belongs in this document more than anywhere: "A 256-bit hashlock gives about 128 bits of Grover preimage margin; HASH160 does not." The older HASH160 form is demoted to recovery of existing locks. This was a post-quantum hardening of the escrow primitive, done quietly, in the same release.

The second is a name. In that same commit the descriptor parser gained a second accepted spelling of the same function:

```
if (Func("htlc_sha256", leaf_expr) || Func("model_htlc_sha256", leaf_expr)) {
```

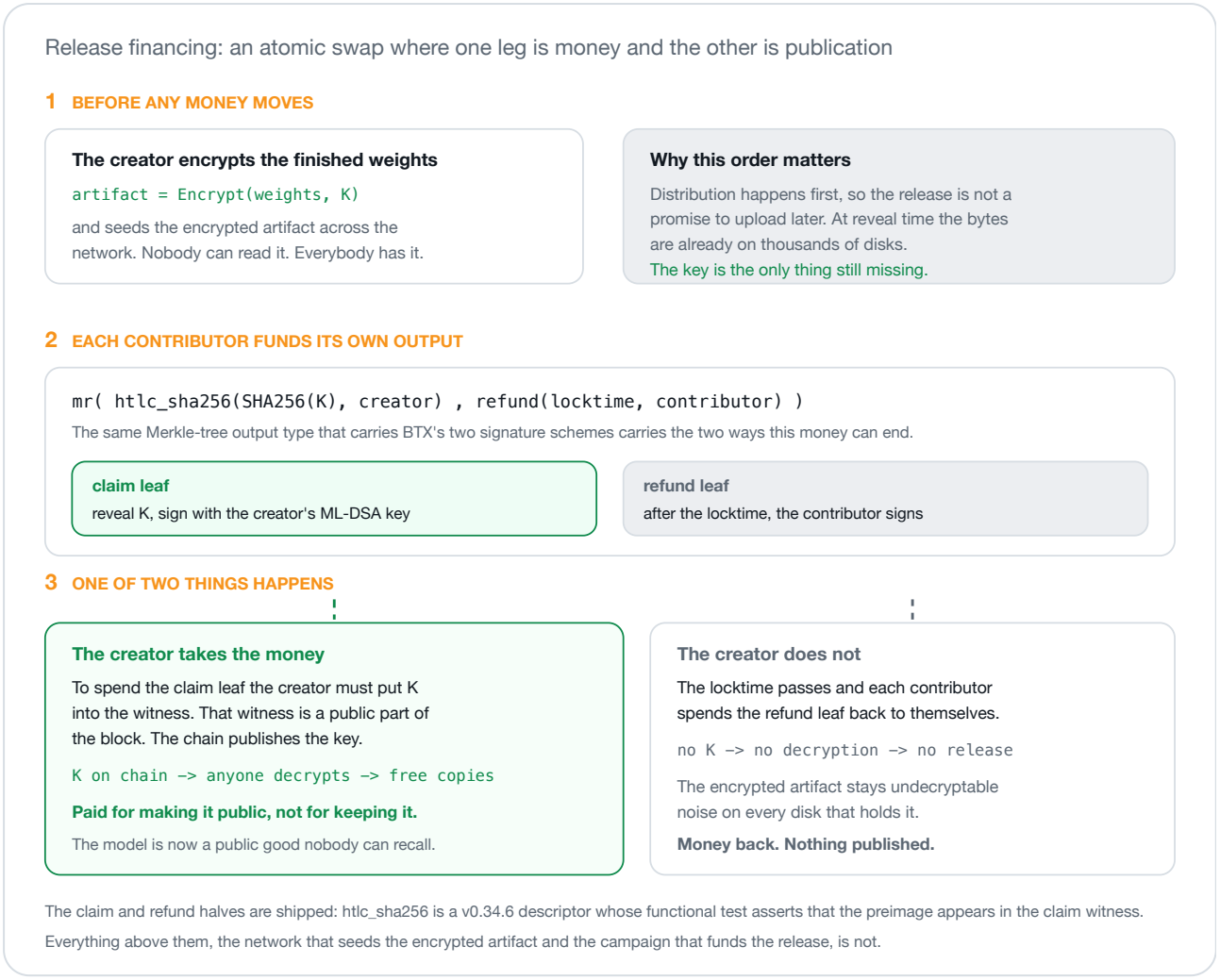
`model_htlc_sha256` is an alias. It parses to an identical script and renders back as `htlc_sha256`, and a unit test named `mr_descriptor_model_htlc_sha256_alias_canonicalizes` asserts that the alias string never survives. It appears in exactly two files in the public tree, and `git log -S` dates both to commit `3013c2c2`.

So the release published that morning contains the model network's financing hook, named after the thing it was built for, hours before the model network was described. For one day it was the only public trace of 0.34.7 that existed. It is also not groundwork laid over months: the SHA-256 hashlock and its alias were authored the same morning they shipped. Nothing resembling them is in v0.34.5.



Redrawn on 14 September, when the middle column stopped being a set of claims about one machine and became code anyone can fetch. The left column is still the only tagged release. The right column is measured from the branch rather than quoted from the release notes, which matters for two of its rows: the acceptance matrix the release notes describe as not run is not in the repository at all, and the phrase monetary load isolation appears in none of the 246 changed files.

9.4 Release financing, and the 1999 paper it rebuilds



Drawn from the 13 September announcement and revised after publication. The claim and refund leaves are tagged in v0.34.6, and since 14 September the funding path that builds, signs and broadcasts is readable in pull request 156 as well. What is still absent from any public tree is the economy around it: the encrypted artifact, the network that would seed it, and any mechanism that gathers many contributors into one contract. Each contributor funds a separate output against the same hashlock, with no threshold and no joint refund, which is what section 9.5 means when it says this is not an assurance contract.

The flow has one ordering decision that carries the design. The creator encrypts the finished weights and seeds the **encrypted** artifact before any money is raised. That ordering turns a promise into a mechanism: the bytes are already on many disks and only the key is missing. Contributions are then locked against the hash of that key. To take the money the creator must put the key in the claim witness, where it becomes a public part of a block, and anyone can read it off the chain and decrypt the copy they already hold. If the key is never revealed, the locktime passes and contributors take the refund leaf.

The creator is paid precisely at the moment the model stops being theirs alone.

The idea is not new, and saying so makes it stronger. In 1999 John Kelsey and Bruce Schneier published the Street Performer Protocol in *First Monday*, having presented it at a USENIX electronic commerce workshop the year before. Their mechanism: the public places donations in escrow, released to an author on condition that the promised work is placed in the public domain. It is the

busker's hat applied to things that are expensive to make and free to copy. Twenty-seven years later nobody has written a better description of the problem open-weight builders now have.

What Kelsey and Schneier could not remove was the escrow agent. Somebody must hold the money and somebody must decide whether the work was really released, and that somebody is a trusted third party with the usual failure modes.

The BTX proposal's one real contribution is to delete that role. There is no adjudicator: release and payment are the single act of spending one output. That is the part that is genuinely novel, and it is worth stating precisely, because almost everything else in the announcement has prior art. Content-addressed model identity exists in Sigstore's model transparency work and in Hugging Face's own content-addressed storage backend. Peer-to-peer weight distribution was done by Mistral in 2023 with a magnet link. The new thing here is not `btx://`. It is that the condition is enforced by arithmetic instead of by a judge.

9.5 Four things the mechanism does not do

Before the four, one correction this document owes its readers, because publication settled a question this part had answered the other way. Writing about v0.34.6, this article said the escrow shipped but the economy around it did not. That was right then and it is wrong now. The pull request implements the funding path on the node: one call freezes a quote and builds an unsigned transaction, a second signs it, and a third broadcasts it and records it in the wallet. There is a functional test that mines one on a private regtest chain. A wallet-funded, chain-settled payment into a hash-locked escrow is working code, not a description.

The trap, and it is worth naming because the pull request's own summary walks into it, is that those three calls are advertised as returning "not implemented". They do, in the separate helper process, and the refusal message says why: the funding calls live on the node instead. A reader who takes the summary at face value would conclude no payment code exists. It does.

What arrived is the payment. What did not arrive is the economy around it, and that is the part the four items below are about. They were written from an announcement. They now describe published code, and none of them has moved.

It is not an assurance contract. The shipped `buildhtlcclaim` takes a single outpoint, and the published pull request adds no script primitive at all. The funding record confirms the shape in code rather than in prose: one claimant key, one refund key, one amount, and a campaign target that lives in a local file with no on-chain binding. So a campaign with many contributors is many independent outputs sharing one hashlock: no threshold, no provision point, no all-or-nothing refund. That is a plain voluntary-contribution game, not the Bagnoli and Lipman construction that makes threshold pledges work. And the free-riding is worse than usual, because the first claim publishes the key to everybody. A contributor who waits receives exactly what a contributor who paid receives. A mechanism for funding public goods that rewards waiting has not solved the public-goods problem; it has restated it on a blockchain.

Nothing binds the ciphertext to the weights. The hashlock guarantees that money and a preimage move together, and that the preimage decrypts one specific file. It cannot guarantee that

the file is the model that was promised, or that the model is any good. The general result here is Pagnia and Gärtner's: fair exchange of two items is impossible without a trusted third party. A hashlock gives atomic release of a secret. It does not give fair exchange of a model, and no amount of script will change that.

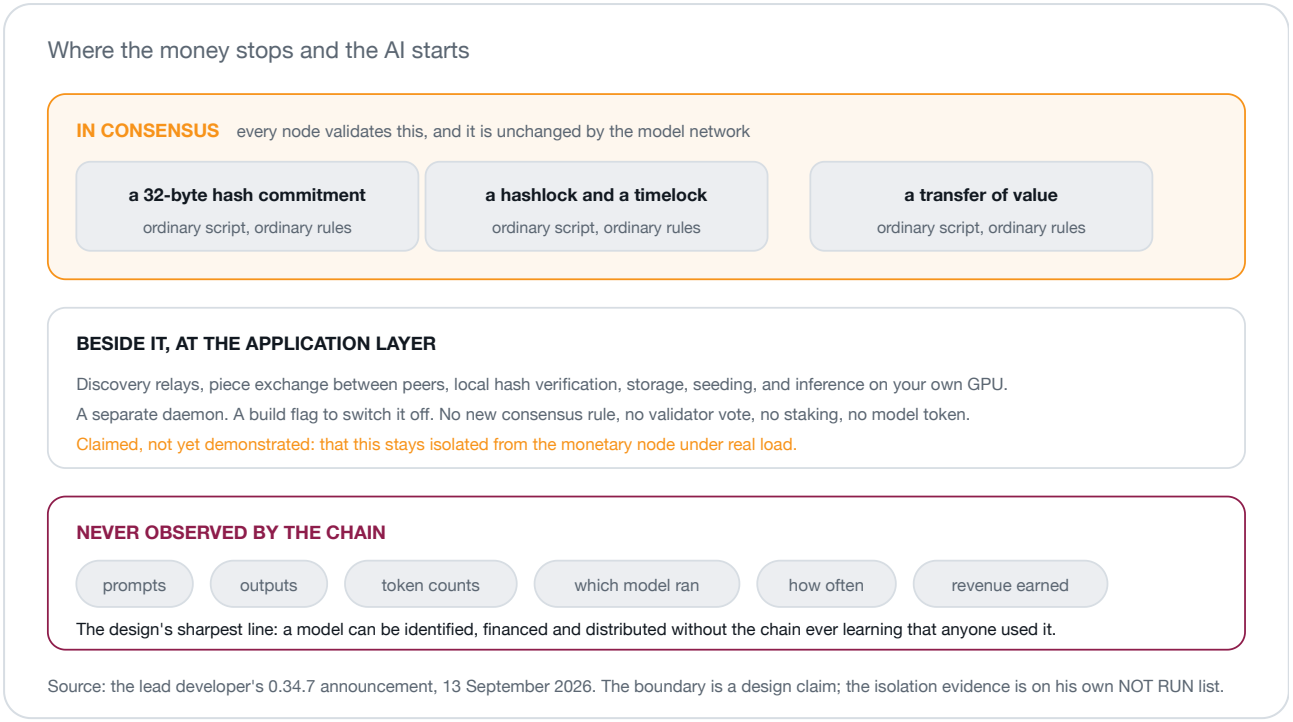
The shipped tooling will not express the full design. The claim RPC in v0.34.6 rejects any P2MR tree that is not exactly two leaves. A third leaf binding campaign terms, or a structure batching many contributors, cannot be built with what exists today. Either new tooling ships with 0.34.7 or the mechanism runs off-chain with the on-chain part reduced to the simplest possible case.

It is velocity, not holdings. The only BTX the design sequesters is flow multiplied by escrow duration. This document has been sceptical elsewhere of token designs whose demand is really throughput, and the same scepticism applies here. There is also a tension inside the announcement itself worth noticing: its best feature is that participation costs nothing, and its stated ambition is that holders will fund releases. A design deliberately engineered to minimise how much money must move is an odd foundation for an argument about holding the money.

This document said in September that either new tooling would ship with 0.34.7 or the mechanism would run off-chain with the on-chain part reduced to the simplest possible case. The second is what happened: the published pull request adds nothing to script, and its own documentation opens by saying it adds no new claim or refund primitive. It also said a campaign with many contributors would be many independent outputs sharing one hashlock, with no threshold and no all-or-nothing refund. The code is built exactly that way, and the threshold that does exist is a number in a file on one machine.

There is one more, and it is a live one rather than a theoretical one. Pull request 137, public in this repository, records that consensus-level script findings remain open and are tracked separately, naming "ML-DSA signature non-canonicality/malleability" and "CSFS signature replay across contracts sharing a hashlock+key", and noting that they touch the script interpreter. Read that second item against the design in this part. A release campaign with many contributors is, precisely, a set of contracts sharing a hashlock and a key.

9.6 Where the money stops, and what would prove it



Sources: the 0.34.7 announcement and implementation report, 13 September 2026. The upper band is what the proposal leaves untouched, which is the whole point of it, and the lower band is what the design states the chain never observes. The evidence gap sits in the middle: the single isolation check the report ran tests btx-tx, which shares base archives with the node but is not where a node-side hook would live, and monetary load isolation is on the report's own list of things that were not run.

The architectural claim is the most defensible thing in the announcement. Nothing about AI enters consensus. Validating a block still means checking a hash commitment, a hashlock, a timelock and a transfer of value. No node is asked to judge whether an inference happened, whether a model is good, or who deserves a reward. No model token, no staking, no validator vote, no inference marketplace. The developer states the boundary as a list of things the chain deliberately never observes: prompts, outputs, token counts, which model ran, how often, and whether it earned anything.

What each design puts inside consensus

design	what consensus must judge	who runs the model	what breaks if it is wrong
Inference as the work	that an inference was really performed	a miner	the money itself
Compute marketplace	that a job was done, and priced	a rented machine	the money itself
Model registry token	which models or providers deserve reward	a staker or validator	the money itself
BTX 0.34.7 as proposed	nothing about AI at all	you, on your own GPU	a file transfer

The taxonomy is by architecture, not by project quality. A design that puts AI judgement inside consensus can still be excellent; it simply makes the soundness of the money depend on the soundness of an AI claim. BTX's stated choice is to refuse that coupling and keep the two failure domains apart.

Four ways a chain can attach itself to AI, sorted by what each puts inside consensus. The taxonomy is about architecture, not project quality: a design that asks consensus to judge an AI claim can still be excellent, but it makes sound money depend on a sound AI claim. The stated BTX choice is to refuse that coupling, so a failure in the model network costs a file transfer rather than the ledger.

Set against the common shape of AI-and-blockchain designs, which put the AI claim inside consensus so that sound money depends on a sound AI claim, this is the more conservative choice, and this document's view is that it is the right one. If the model network breaks, a file transfer fails. The ledger does not.

That is the claim. The evidence for it is thinner than the claim.

This article said in September that the report's one isolation check tested the wrong binary, and the published code confirms it precisely. The build links the model library into the node and into nothing else, which is exactly why the check comes back clean and proves nothing, and the node-side hook this document predicted is there: the peer-to-peer message handler gains three new message types and the protocol gains two service bits. The same paragraph warned that a second implementation of ML-KEM-768, arriving as a new runtime dependency for a daemon that holds money, deserved to be argued in a pull request rather than assumed. It now is one. The node takes on that dependency by default, because the build option that carries it defaults to on.

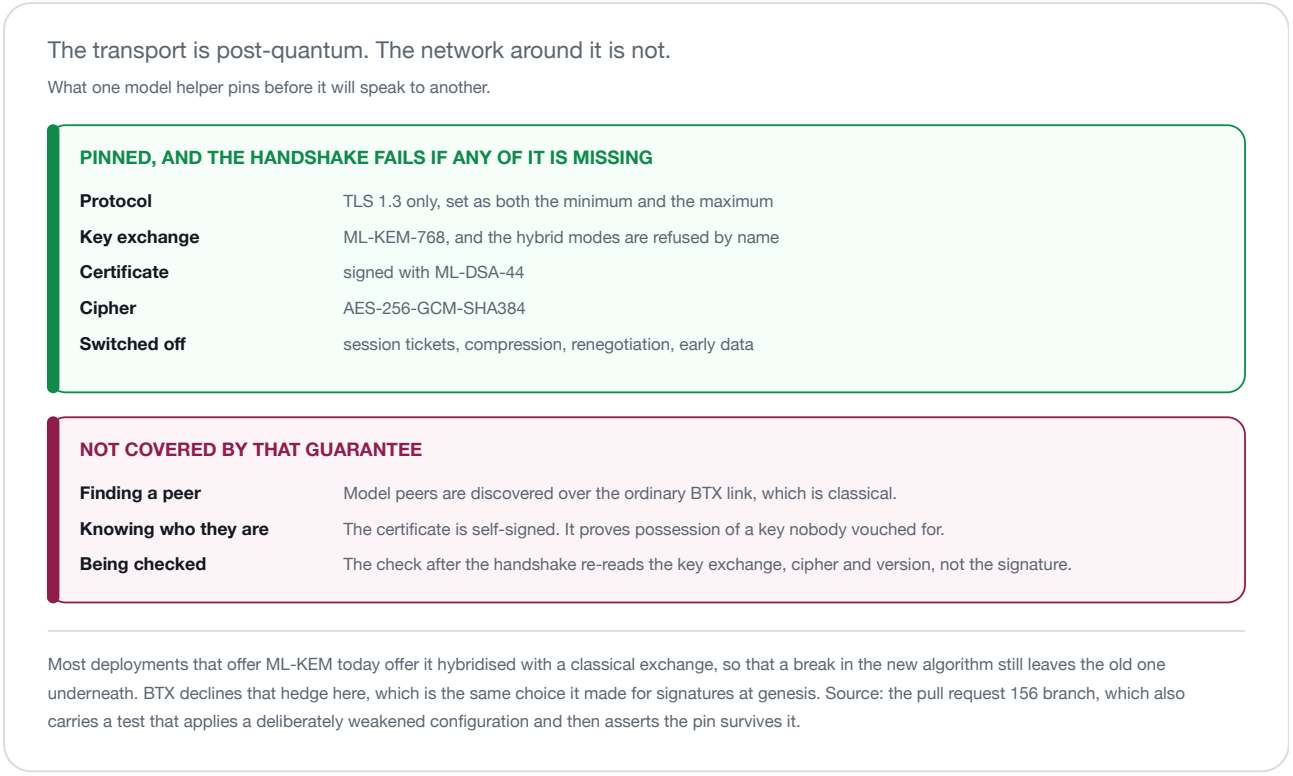
The original sentence, kept because the reasoning is what matters: the report's one isolation check is that `btx-tx` contains no model-network symbols. That tests the wrong binary. `btx-tx` shares its base archives with the node, so a clean result rules out one route and says nothing about the node binary itself, which is where a node-side hook would live. The measurement that would settle it, "monetary load isolation", is on the report's own list of things not run, alongside live multi-node relay and paid retrieval.

The detail underneath that concern, for a reviewer: the handshake needs system OpenSSL 3.5 or newer, and the reported one ran on 3.5.5. BTX did not previously depend on OpenSSL at runtime and already vendors its own ML-KEM-768, so the node now carries two implementations of the same primitive from two sources.

Size is the one number in the private report that did not survive publication, and this article got it wrong by repeating him. The report said 82 files and about 7,500 added lines, and this document used that to argue the change was an ordinary magnitude of work, because pull request 135 had been 78 files and about 6,600 on the same day. The published diff, as the pull request opened, is 240 files and 33,367 added lines, and it has grown since. That is three times the files and five times the lines of pull request 135, and roughly four and a half times what the report described. Either the work grew substantially in the day between the report and the pull request, or the report was measuring something narrower. Nothing in the public record settles which.

What replaces the old concern, that nobody outside one machine could read the code, is narrower and still real: What replaces it is narrower and still real: thirty-three thousand lines arrived in a single pull request, against a repository where, as Part 10 measures, no human being has ever approved one.

9.7 The transport, which is the genuinely new cryptography here



Read from the branch rather than from the release notes. The upper band is what the transport pins; the lower band is what that guarantee does not reach. The branch also carries a test that applies a deliberately hostile configuration, permitting an older protocol and a classical exchange, and then asserts the pin still holds, which is a stronger property than simply setting it.

Everything this document has said about post-quantum cryptography so far concerns signatures, because that is what consensus uses. The model network introduces the other half of the problem: it is the first place BTX depends on a key encapsulation mechanism on the wire rather than inside a transaction.

Two helpers will not talk to each other unless the connection is TLS 1.3, the key exchange is ML-KEM-768, the certificate is signed with ML-DSA-44 and the cipher is AES-256-GCM-SHA384. Session tickets, compression, renegotiation and early data are all switched off. None of that is a

preference that degrades. The pin is set on the context before a connection is attempted, and a peer that cannot meet it fails the handshake.

The choice worth pointing out to a reader who follows this field is the refusal of hybrids. Most deployments that offer ML-KEM today offer it combined with a classical exchange, so that a break in the newer algorithm still leaves an older, well-studied one underneath. The BTX code refuses those combined modes by name, in a comment that names them. That is the same bet the chain made at genesis when it declined to keep an elliptic-curve path alongside the post-quantum one, now made again one layer down. It is coherent, and it is a bet rather than a free improvement.

Three things stop this from being a claim that the model network is post-quantum. A helper learns about other helpers over the ordinary BTX peer-to-peer link, which is classical. The certificate at the other end is self-signed, so it proves possession of a key that nobody has vouched for, and a helper serving a file does not pin its callers at all. And the check performed after the handshake re-reads the key exchange, the cipher and the protocol version, but not the signature algorithm, which is the half a reader of this document would care about most. The accurate summary is narrower and still notable: the hop that carries the bytes is post-quantum and fails closed. Finding the peer and knowing who it is are not.

One practical note, and it changed while this revision was being written, which says something about the pace here. The transfer path works around a wide-area network problem by capping record size, and the call it originally used exists only on Linux, so as the pull request was opened there was no path to this subsystem from a Mac. A fifth commit landed at 14:23 UTC on 14 September adding one: a Darwin branch for that workaround, a statically linked build, and a packaged Apple Silicon target. The pull request this part describes is three hours older than the sentence you are reading, which is the ordinary condition of writing about this project.

9.8 How to check this yourself

Four commands settle today's state, and none of them needs our word for anything.

```
# 1. Is there a 0.34.7? A message in a group chat is not a release.
gh api repos/btxchain/btx/releases --jq '.[0].tag_name'
#    v0.34.6

# 2. The financing hook that did ship, under the name it was built for.
gh api repos/btxchain/btx/contents/src/script/descriptor.cpp --jq .content \
  | base64 -d | grep -n model_htlc_sha256
#    if (Func("htlc_sha256", leaf_expr) || Func("model_htlc_sha256", leaf_expr)) {

# 3. The tree that was private on 13 September. This is the exact commit the report
named.
gh api repos/btxchain/btx/commits/b094c6ba420f1e7cb277a586c35aae806d59a8c5 --jq
.commit.message
#    Vendor v1.1 evidence vectors and JSON schemas for the copied Python reference.

# 4. The pull request that carries it. The size moves, so this prints today's.
gh pr view 156 --repo btxchain/btx --json state,changedFiles,additions
#    OPEN, and larger than the 240 files it opened with
```

This article listed five things worth checking when 0.34.7 appeared. Here is the scoring.

1. **A tag and a pull request**, with a diff a stranger can read. Half met. The pull request exists and the diff is readable. There is no tag, and the branch declares itself not a release.
2. **Whether consensus files moved.** Passed, and this is the important one. Nothing under `src/consensus/` moved, and neither did the validation logic, the script interpreter, the chain parameters, the proof-of-work code or the mempool policy. The twelve deleted lines in the entire change are an edit to one test harness. The architectural claim this part treated as a claim is now a fact about a diff.
3. **Whether the model network compiles out**, and whether the monetary binaries are identical when it does. Half met. It does compile out, through a build option, and the pull request ships a script to demonstrate it. That script says in its own header that it never builds the node, so the second half of the question, whether the monetary binaries come out identical, is still unanswered. The option also defaults to on, so the ordinary build is the model-network build.
4. **Whether the unrun list has been run**, particularly monetary load isolation. Not met, and the gap is wider than the release notes suggest. That phrase appears in none of the 246 changed files. The acceptance matrix the notes report as not run is not in the repository at all, so it cannot be read either way. The repository runs no continuous integration, so none of the tests in this change were executed by a machine when it was opened. And two scripts that look like evidence, including one reporting a fourteen-gigabyte transfer, print fixed text rather than measuring anything.
5. **Whether anyone outside the project has read it.** The MatMul v4 fork went to an external reviewer who returned a NO-GO and a remediation map before activation, and the project published both. That is the standard this change should meet, and the project set it itself.

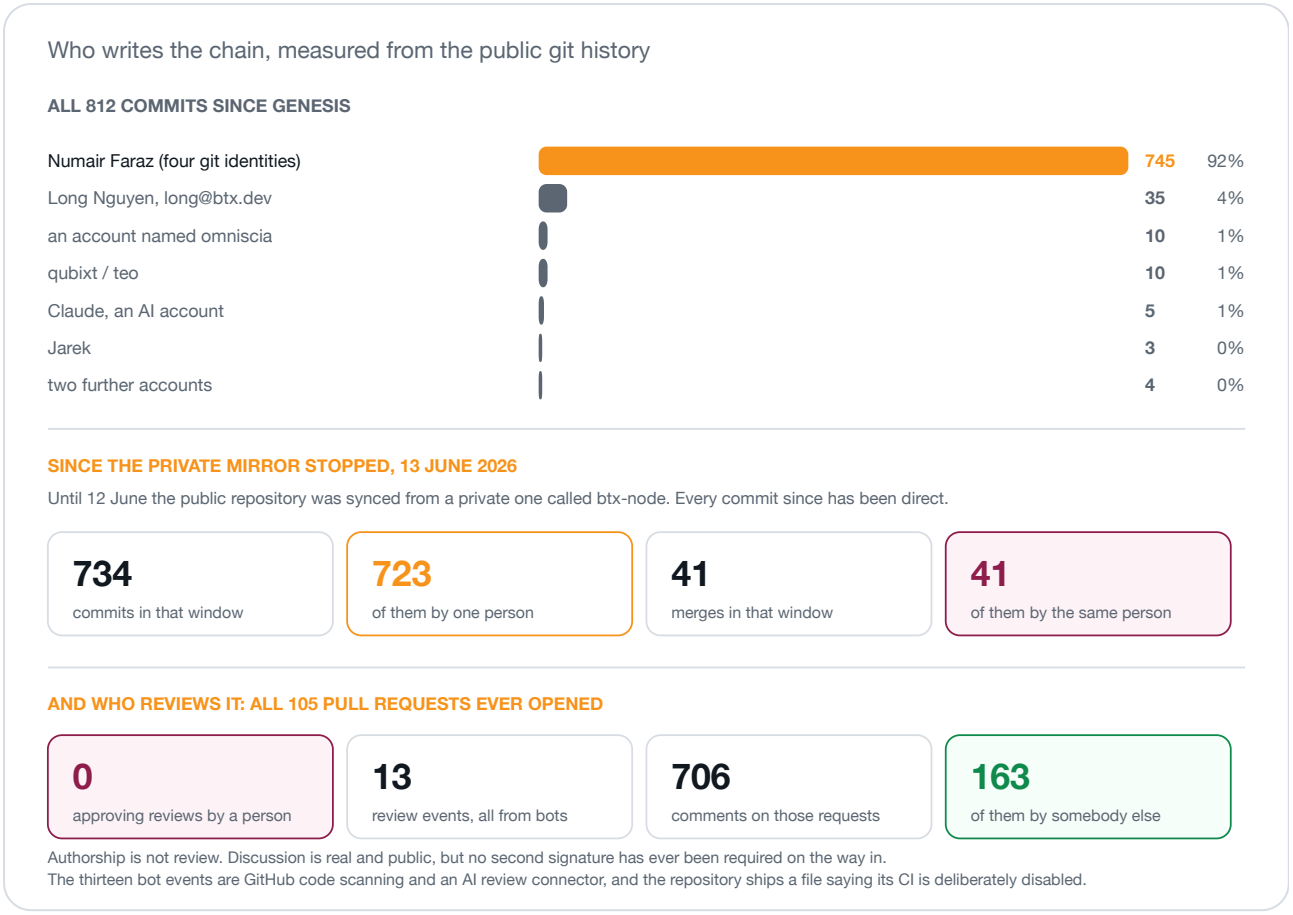
A last word about pace, which is not an argument against ambition. This chain published a difficulty rule for block 199,299 on 25 August and withdrew it two days later, and the nodes that had taken it sat parked below the boundary while the rest of the network carried on. Part 5 records that week. A release described as arriving "in the next 24 hours", carrying several thousand new lines into software that settles money, is a moment to read the diff rather than the announcement.

Part 10. Who decides, and what happens next

Who makes BTX do anything is the question a serious reader asks last and cares about most. There is a blunt version of it that circulates in the group chat: if the developer were hit by a bus, does BTX die?

The answer is not one word, and it is worth separating into layers, because the layers behave very differently. Everything below is measured from the public repository and the public chain, and every number in it can be reproduced by anyone with a clone and a terminal.

10.1 The bus factor, measured



Counted from a clone of the reference repository on 13 September 2026, so anyone can reproduce it. The four identity figures are git author counts across four email addresses that resolve to one GitHub account; the review figures come from the GitHub API across every pull request the repository has ever had. The thirteen bot events are GitHub's code-scanning bot and an AI review connector. There are also zero review trailers of any kind in the 812 commit messages.

The tree holds 812 commits. One person wrote 745 of them, a little under ninety-two percent, under four git author addresses that all resolve to the same GitHub account. On the files where mistakes cost money, the consensus directory, the validation logic, the script interpreter and the chain parameters, the concentration is the same: 177 touches out of 198.

Merges are the more interesting number, because merging is the act of deciding. Across the whole history, 51 of 71 merges are his, which leaves room for the reasonable thought that somebody else is also steering.

Until 12 June 2026 the public repository was not where BTX was developed. It was a mirror. The commit titles say so in plain language, "Sync public btx with btx-node main for BTX 0.32.1" and "Sync public repo from btx-node for v0.32.3", and every one of the seventeen merges by the second-most-active account is one of those syncs. A second engineer was keeping a public copy in step with a private repository called `btx-node`.

That stopped on 12 June, and the change is genuinely good news: development moved into the open, where pull requests and diffs are visible as they happen. It also removed the only other person who

was merging. In the 734 commits since, 723 are his, and of the 41 merges, 41 are his. Not most of them. All of them.

The rest of the contributor list is short and worth naming, because a fair account of key-person risk includes the people who are actually there. Long Nguyen has 35 commits, every one a mirror sync, the last of them on 12 June. An account named omniscia has 10, four of them fuzzing harnesses for the post-quantum signature library and the rest build and tooling work on the Apple Metal path, written across April and May. That name invites a natural assumption and it should be headed off: it is a personal GitHub account with no name, company or biography set, and it is not the audit firm called Omniscia, whose own presence is a separate organisation account. Nothing in the BTX tree references that firm, and the caveat this document has carried from the beginning stands unchanged: no named third-party firm has signed off on this stack. The project says the same thing about its own audit documents, one of which states in its title line that it is not a signed independent human engagement. An account under the name qubixt has 10 commits, a Claude AI account has 5, a contributor named Jarek has 3, and two further accounts have 2 each. Those seven names and 67 commits are the whole of the rest.

One number needs correcting against a naive reading, including the one this document carried in earlier revisions. GitHub's contributor graph shows a much smaller figure for the `numair` login, because it counts only commits whose author email maps to that account, and most of his work is committed under an address that does not. The git history is the honest source, and it says 745.

Authorship is not review. A project can have one prolific author and still be safe if other people read the work before it lands.

Across all 105 pull requests ever opened on the reference repository, 56 of them merged, the number of approving reviews is zero. Not a low number. Zero. The only review events of any kind are thirteen automated comments: seven from GitHub's own code-scanning bot and six from an AI review connector. The commit messages say the same thing from the other direction: across all 812 commits there is not one acknowledgement, reviewed-by or signed-off-by trailer. No human being has ever formally approved a change in this repository.

Two things keep that from being the whole story, and both deserve to be said. Pull requests do get discussed: 706 comments across those 105 requests, from seventeen accounts, sixteen of them people, and 163 of those comments come from somebody other than him. Outside scrutiny is real and substantive. It simply arrives in a different shape than the approval count measures, as filed issues and contributed test code rather than as review, and it lands when he picks it up and applies it. What the zero does establish is narrower and still important: nothing reaches this codebase that one person has not decided to put there. The gate has one hand on it.

10.2 What survives if he stops

If he stopped tomorrow, what breaks

The answer is not one word. It differs by layer, and the project's own documents say so.

KEEPS RUNNING WITHOUT HIM

Your coins and the ledger	Ordinary UTXO rules. No key of his can move or freeze them.
Block validation	No key in the tree governs consensus. A node accepts a block because it replayed the work itself.
Mining	Admission is a byte-exact self-test on your own hardware, not a list he controls.
Observed, not theorised	The network produced and accepted blocks to 199,328 while both operator nodes sat at 199,300.

DEGRADES WITHOUT HIM

New nodes joining	Three DNS seeds, one address each, all inside one hosting provider.
The DNS fallback	Two compiled-in addresses, and both are already two of those three.
Nodes that follow a pin	Trusted-mirror nodes advance only while the keys they pinned keep signing.
The update channel	On by default, one operator URL, one compiled-in key. Its endpoint is months stale.

STOPS WITHOUT HIM

Releases, fixes, security patches	41 of 41 merges since June 2026 were his.
The roadmap	Every undated item, and the model network, exists only where he works.
Merging and tagging	0.34.7 is readable as pull request 156. Only he can merge or tag it.

The seed gap is not an outside criticism. The comment sitting above those two entries, in the shipped source, says in as many words that a two-entry subset of the same DNS set is not a fallback. The update channel is the one power over nodes the README does not list.

Sources for each band, in order: the ordinary UTXO rules; the absence of any attestation signer key in the tree; the mining self-test described in Part 8; the blocks the reference repository records as accepted to 199,328 while both operator nodes sat at 199,300; src/kernel/chainparams.cpp and src/chainparamsseeds.h for the seeds; and the README's own trusted-mirror notice. The middle band is the one the project documents against itself, and section 10.2 quotes the comment.

The chain does not need him. This is the strongest claim in the project's favour and it holds up under inspection, . No key anywhere in the tree has authority over consensus: there is no compiled-in attestation signer, and searching the whole source for embedded public keys turns up nothing that a block is checked against. A node accepts a block because it replayed the work itself and got the same digest, not because anyone authorised it. Mining admission is a byte-exact self-test on your own hardware rather than a list of approved devices. Your coins sit behind your own post-quantum keys, and no key of his can move or freeze them. The project offers evidence for this that is better than an argument, because it actually happened: during the late-August disruption the network produced and accepted blocks up to 199,328 while both operator-run nodes sat at 199,300. Consensus formed without them. One qualification belongs with it, in fairness to the reader rather than to the project: those twenty-eight blocks took four hours and fifty-one minutes, roughly ten minutes a block against a ninety-second target. The network kept going without its operator, and it was limping while it did.

Joining the chain does need him, for now, and the source says so plainly. A node that already has peers keeps running. A node starting from nothing has to find somebody, and the addresses it is built to ask are all his. Mainnet ships three DNS seeds, at `node.btx.dev`,

`node.btxchain.org` and `node.btx.tools`. Each resolves to exactly one address today, and all three of those addresses sit inside a single hosting provider's network.

The fallback is thinner than the seeds. When DNS fails, a node falls back to a table of fixed addresses compiled into the binary. That table has two entries. Decoded, they are the same machines that `node.btx.dev` and `node.btx.tools` already resolve to.

The reason to state this so precisely is that the project states it first. The comment sitting directly above those two entries reads, in the shipped source: fixed seeds must be regenerated from a crawl with diverse network operators, and a two-entry subset of the same DNS set is not a fallback. The comment above the DNS seeds sets the target at six to eight seeds across distinct networks. So this is not an outside criticism. It is a known gap, written down by the person who left it, in the file where it lives, and it is the single most concrete piece of centralisation remaining in a design that has worked hard to remove the rest.

Nodes that follow a pin need somebody to keep signing. A machine without a qualifying GPU cannot replay the work itself, so it follows signed verdicts from nodes that can, under keys its operator chose. The README is direct that after v0.34.1 the original operator no longer commits to running those signers, and that a node whose pinned keys stop signing stops advancing. The project's own recommended fix is the right one: stop delegating and validate for yourself.

And one compiled-in key can ship code to default nodes. This is the one place where the README's own wording reaches further than its evidence. It says there is no compiled-in signer key anywhere in the tree, and offers a search of the chain-parameters file as proof. For consensus that is correct. For the tree it is not. A separate file compiles in a release key, an ML-DSA-44 public key more than two thousand characters long, and the auto-updater that uses it is on by default on mainnet. Left alone, a node polls one operator-controlled address every thirty minutes and, by default, launches the installer for any newer signed release it finds.

Four things make that far less alarming than it first sounds, and all four are real. The private half is stated to be offline, so compromising the website, its DNS or its certificate is not enough to push code. The installer script must match a hash in the signed manifest. The updater refuses anything that is not newer than what is running. And any operator can override the key or switch the whole mechanism off. The design is careful, and choosing a post-quantum signature for the update channel on a post-quantum chain is exactly the right instinct.

What it is not is absent from the list of things that depend on one person. It is the single key in the repository with power over what nodes run, and it belongs in an honest account of key-person risk. There is also a smaller fact that says more about the handover than any of the prose does. That update endpoint is live, and today it still advertises version 0.32.8 from a commit dated 13 June 2026, while the current release is v0.34.6. Roughly twenty releases have shipped past it. Nothing bad happens, because the updater will not go backwards. But one of the operator-run legs the README lists as still theirs has quietly stopped being maintained, and the shipped binary still points at it.

The software stops. Releases, security fixes, the response to the next stall, and every unfinished item on the roadmap live where he works. That is not a criticism of his output, which has been

prodigious by any standard. It is a description of a single point of failure, and it is the layer where the bus question has a real answer.

10.3 The handover that has not happened yet

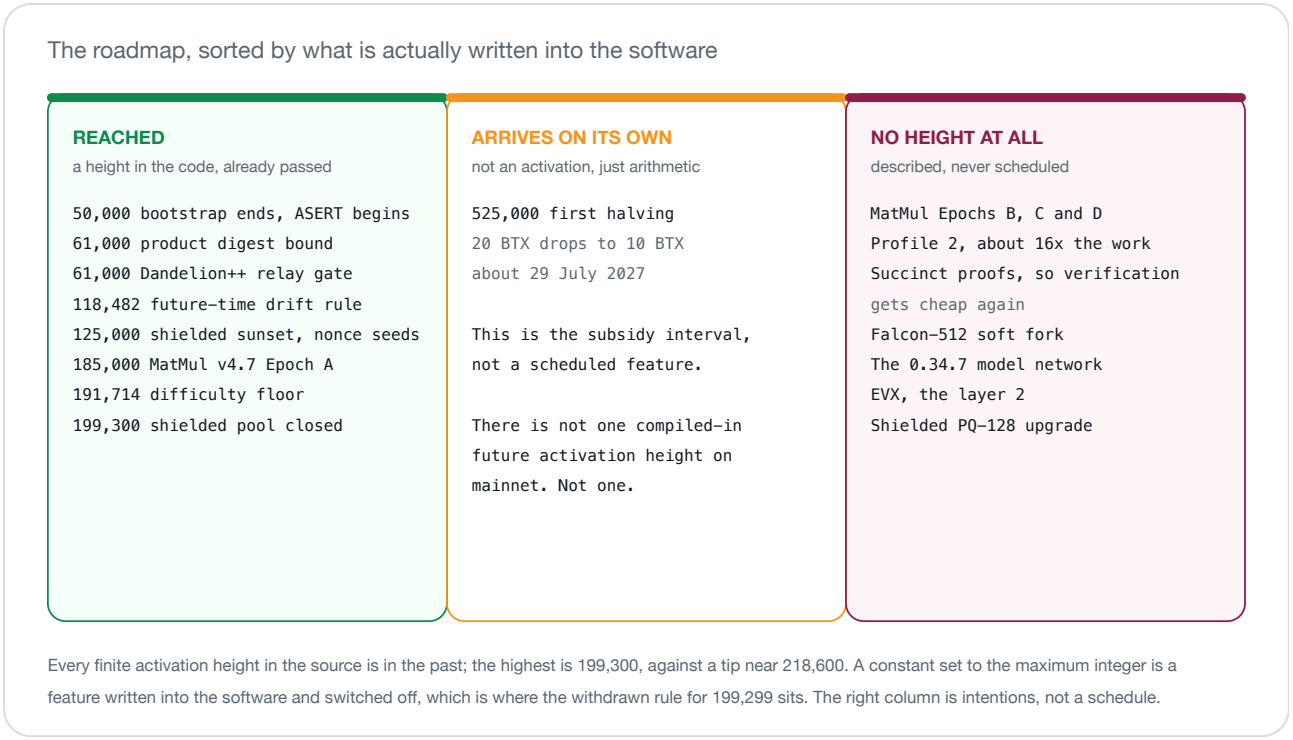
The project has thought about this harder than most, and has written two documents specifically so that somebody else could continue: one on how a fork produces its own mining-admission golden reference and reseals against its own freeze without asking permission, and one on how a third party cuts a release without the original machines or keys. Those documents are real, they are specific, and their existence is to the project's credit. Most projects with this shape have nothing of the kind.

The README goes further and declares the matter settled. It says v0.34.1 is the base reference implementation, that further enhancements, features and releases are expected to come from community forks and modifications rather than from that repository, and under its own heading, that the team is stepping back.

That declaration was committed on 27 August 2026, alongside the v0.34.1 tag, and the README was last touched on 31 August. In the seventeen days since, the same repository has published v0.34.2, v0.34.3, v0.34.4, v0.34.5 and v0.34.6, the last of them on 13 September, and has described v0.34.7 as arriving within a day. Two hundred and three commits landed after the handover was declared, two hundred of them his, and the most recent release changed seventy-eight files including the validation logic. The README still calls v0.34.1 the current release.

It would be easy to call that hypocrisy, and it is not. Every one of those releases fixed something real, and several were direct responses to a network that needed them. Stepping back from a chain mid-incident would have been the worse choice. The honest reading is simpler and more useful: the handover is an intention that the facts have not caught up with. BTX today is a project with a stated succession plan, genuine technical groundwork for it, and no succession. A reader deciding what to weight should weigh the plan and the practice separately, and notice that so far only one of them exists.

10.4 The roadmap, sorted by what the software agrees to



Every activation height compiled into src/kernel/chainparams.cpp, sorted by whether the software has agreed to it. Every activation height compiled into src/kernel/chainparams.cpp, sorted by whether the software has agreed to it: the past, the halving, and the undated. Measured against a tip near 218,600.

There is a long-term plan here, and it is more coherent than most. It is also almost entirely undated, and the gap between a plan and a schedule is visible directly in the source, which is the useful thing about a chain: intentions live in documents, but commitments live in constants.

Activation heights are compiled-in numbers, so a reader can simply look at them. Eight are in the past, from the end of the bootstrap at 50,000 through the Dandelion++ relay gate at 61,000 and the MatMul v4.7 switch at 185,000, to the shielded pool closing at 199,300.

Then the list stops. Against a tip near 218,600, there is not one compiled-in future activation height on BTX mainnet. Not one. The only future event written into the software is the first halving at block 525,000, which is not an activation at all: it is the subsidy interval doing arithmetic, dropping 20 BTX to 10 BTX somewhere around the end of July 2027.

Everything else is either a constant set to the maximum integer, which is the software's way of saying a feature exists and is switched off, or it is prose in a design document with no constant at all. MatMul Epochs B, C and D have no heights. Profile 2, the workload roughly sixteen times the current one, has no height. Succinct proofs, which would make verification cheap again and undo the concession described in Part 8, have no height. The Falcon opcodes are reserved and never enabled. The shielded parameter upgrade has sat at maximum since it was written. The withdrawn rule for block 199,299 sits there too, which is what a retracted plan looks like in code.

The model network of Part 9 is not even that, and now for a clearer reason than when this was written. It has no activation height because it is not a consensus feature at all. It is a compile-time

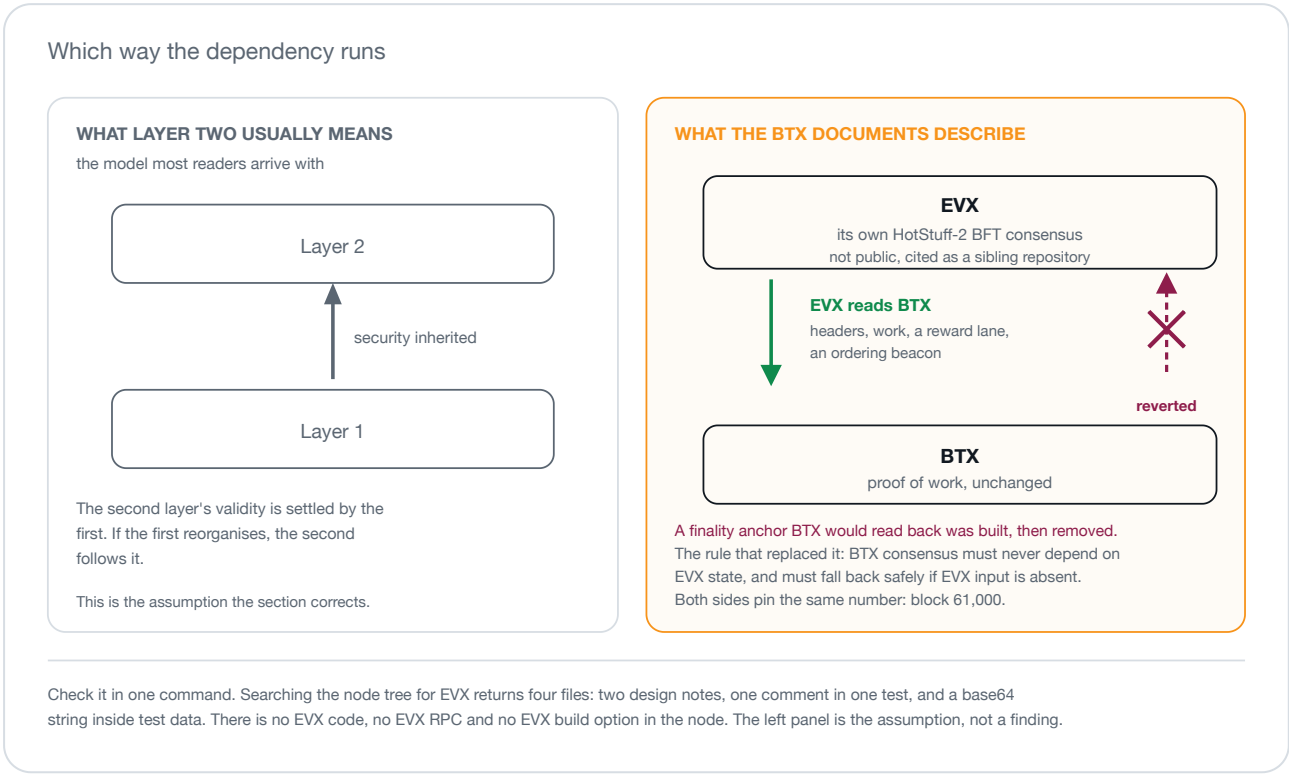
option, defaulting to on, which is a different kind of thing from every item above it. There is a branch and a pull request now. There is still no tag.

There is a tension between this roadmap and the handover, and it should be named rather than smoothed over. If v0.34.1 is the base reference and further releases are expected from community forks, then nobody has said who ships Epoch B. The roadmap quietly assumes a maintainer the handover says is stepping back.

10.5 EVX, and what a second layer here actually means

EVX is the clearing layer the project intends to sit above BTX. It is the most frequently discussed unshipped thing in the ecosystem, and it is also the one most often described inaccurately, including by people who like it. The public BTX repository says more about it than most readers realise, so it is worth separating what is documented from what is expectation.

EVX is not a rollup, and BTX does not secure it. Two design documents in the BTX tree state that EVX runs its own HotStuff-2 Byzantine-fault-tolerant consensus with a stake-fallback beacon, and one of them says the consequence in a single sentence: BTX proof of work is never a dependency of EVX safety or liveness. BTX feeds it a reward lane and an anti-manipulation ordering beacon, and that is the extent of it. If you are used to the Ethereum sense of layer two, the second layer does not inherit the first layer's security here.



The left panel is the assumption this section corrects, not a measurement of any chain. Everything in the right panel above the BTX box is BTX's account of a repository nobody outside can open, so the HotStuff-2 consensus and the six-confirmation deposit floor are quoted rather than read. What can be checked is the absence: searching the node tree for EVX returns four files, being two design notes, one comment in one test, and a base64 string inside test data. There is no EVX code, no EVX remote procedure call and no EVX build option in the node. Sources: doc/btx-evx-finality-coupling.md and doc/btx-op-check-multi-pq-plan.md.

The second thing worth knowing is that the coupling has been tried and deliberately backed out. The same document records an EVX-anchored finality floor that was built and then reverted, and states the rule that replaced it: every BTX-side change must remain node-local policy, must fall back safely if any EVX-derived input is stale or absent, and must never make BTX consensus depend on EVX state. A second decision points the same way. EVX carries a placeholder opcode for post-quantum multi-signature covenants, and the BTX-side analysis concludes that BTX should add nothing for it, recommending instead that EVX build its covenants out of script primitives BTX already has. The preferred path is explicitly the one that requires zero BTX consensus change.

For a reader who cares about BTX as money rather than as a platform, that removal is more convincing than a promise, because it cost something.

Two honest limits. The first is that EVX itself is not public. Both documents reference it as a sibling repository, and the deposit verifier, the consensus node, the validator bond of 1,050 BTX and the covenant configuration they cite cannot be read by anyone outside. There is no EVX whitepaper in public, and btx.dev does not mention EVX, wrapped BTX or a clearing layer anywhere across its English pages. Everything above is BTX's account of EVX, not EVX's.

The second is timing, and it should be attributed to what it is. The project has told its community that no EVX launch should be expected before June 2027, and has framed that as deliberate long-horizon sequencing rather than delay. That is a statement from the team, not a document and not a constant, and this article treats it the way it treats every other statement of intent in Part 9: as a claim about the future made by the people who would have to fulfil it. Nothing in the public tree dates EVX at all.

What does exist publicly is the bridging groundwork, and it is further along than the layer itself. The repository carries a wrapped-BTX toolkit with EVM reference contracts, a federation lock-and-mint bridge, a trustless atomic-swap contract whose hashlock is deliberately BTX-compatible, a Foundry test suite that passes, and a threat model mapping each defence to the bridge hack it is meant to prevent. It also carries, in seven separate files including the contracts themselves, the words that matter most: these are unaudited references, and no real value should move through them before an external audit.

One correction to a number that circulates in the community, and that an earlier revision of this document repeated. A bridge pilot did run, and people quote the few thousand satoshis it wrapped as evidence that the bridge is live. It is not evidence of that. The BTX side of that pilot ran on a local regtest chain rather than mainnet, so no mainnet BTX has ever been locked; every token transfer in it happened on a single day in July 2026; the reserve figure attached to it is a number an operator set by hand and has not updated since; and the only source for any of it is our own explorer. We are disclosing that because we built the explorer, and a figure sourced from oneself is not an independent measurement.

10.6 What this adds up to

The fair summary is neither of the two sentences people reach for.

"It runs itself" is wrong. One person writes the software, one person merges every change, no human has ever approved a pull request, new nodes find the network through three names that resolve into a

single hosting provider, the fallback compiled into the binary is two of those same machines, and one compiled-in key can ship code to every node running the defaults. The handover is written down and has not happened, and the update endpoint it left behind has been stale for three months. If he stopped today, the chain would keep producing blocks and your keys would keep working, and the software would stop moving.

"It fails without him" is also wrong, and it is the more interesting error, because it assumes the part that matters most is the part that depends on him. It is not. Validation, mining admission and ownership are the layers where a chain either is or is not money, and all three are already independent of any person: no signer key exists, every node checks the work itself, and the network demonstrated in August that it forms consensus while the operator's own machines are behind. That property was engineered deliberately, and it is rarer than it sounds.

So the accurate version is layered. The money is decentralised. The maintenance is not. The distance between those two facts is the risk a reader is actually taking, and unlike most risks in this space it is not permanent, because the things that would close it are small, specific and checkable from outside. More seed operators on more networks, which the source file already asks for by name. A second person with merge rights. One approving review on one pull request. A release cut by somebody else using the procedure the project has already written down. An update manifest that is not three months behind the release it points at.

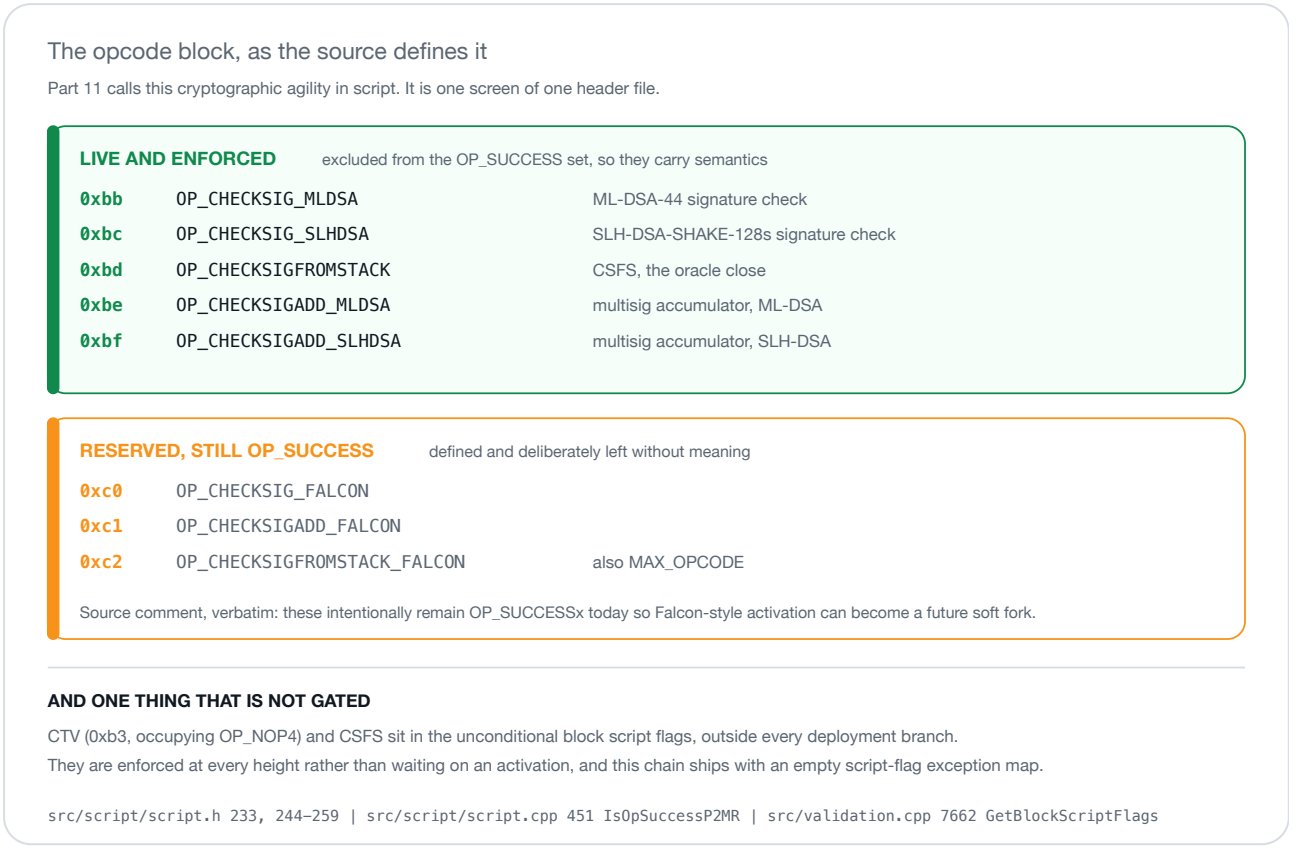
None of that requires a new invention. All of it requires a second person. Until one appears, the correct description of BTX is a chain whose consensus has been carefully built to need nobody, maintained by a project that still needs one.

Part 11. What is genuinely unusual here

The pro column, for someone who reads consensus code.

1. **The quantum migration is finished before it started.** No exposed-key coins, no confirmation-race exposure, no burn-versus-steal governance fight, no soft fork to get through. This is the single hardest thing Bitcoin has ahead of it, and BTX does not have it.
2. **Two-scheme defense with independent assumptions**, always present in every output, plus reserved opcodes for a third. Cryptographic agility in script.
3. **Bitcoin's money rules, untouched.** Same cap, same halving logic, UTXO conservation, no token layer, no admin keys.
4. **Covenants are live.** CTV vaults and CSFS oracle closes are consensus today. On Bitcoin they are still proposals.
5. **Two independent GPU backends must produce identical bytes.** A total replacement of the work function shipped without adding a byte to the header.
6. **A documentation culture that publishes its own NO-GOs**, rejected parameters, stalled forks and unfinished components disabled in code. In six months this project has corrected itself in public repeatedly. That is a better signal than any brochure.

7. **A handover model.** v0.34.1 is declared the base reference; "further enhancements, features, and releases are expected to come from community forks and modifications, not from this repository." The documents explain how a fork adds its own golden, reseals against its own freeze and mines without asking anyone's blessing. Consensus validation depends on no pin, no signer and no key; the network produced and accepted blocks up to 199,328 while both operator nodes sat at 199,300. That is Bitcoin's founding pattern, deliberately repeated.
8. **An escrow that pays for publication.** The hash-locked contract that shipped in v0.34.6 makes a payment and a disclosure the same event, because claiming the money puts the secret in a public witness. That is the 1999 Street Performer Protocol with its trusted escrow agent removed, . Part 9 covers both the idea and the four things it does not solve.



Item 2 above calls this cryptographic agility in script and item 4 says the covenants are live. Both are one screen of one header file, and this is it. Five opcodes are excluded from the OP_SUCCESS set, so they carry meaning today. Three adjacent numbers are left inside that set on purpose, with the reason written in the source comment, so a third signature scheme can arrive as a soft fork rather than a chain split. The two covenant opcodes sit in the unconditional block script flags, outside every deployment branch, which is what makes item 4 checkable rather than merely asserted. Sources: src/script/script.h, src/script/script.cpp and src/validation.cpp in the reference node.

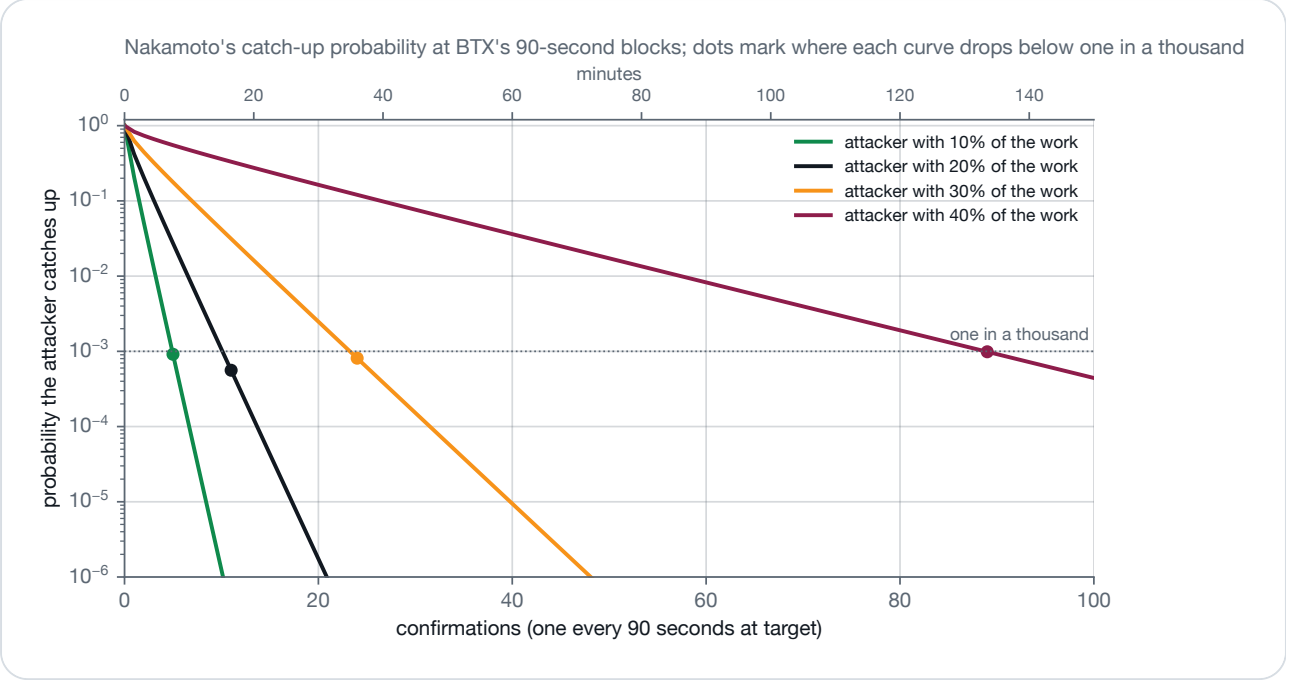
Part 12. The contra column, without cushioning

An experienced reader will find these anyway. Better they come from us.

1. **No market.** There is no exchange, no order book, no price. Any number you see is a model (btxprice.com multiplied BTC's price by a constant and BTX's own hashrate, and it has been frozen since 30 August) or two people agreeing privately. Treat anything mined as participation, not income.

2. **Six months old and small.** Roughly 60 reachable nodes, one or two working pools, a security budget that is a rounding error next to Bitcoin's. On a small proof-of-work chain deep reorganizations are a realistic risk; BTX's own vulnerability assessment says no node-local rule can make a one-confirmation settlement final against a stronger private miner. Wait for confirmations in proportion to value, as the whitepaper's own table suggests (six confirmations, nine minutes, against a 10 percent attacker).
3. **The genesis million.** Blocks 1 to 50,000 were mined at a 250-millisecond cadence in about six and a half hours on 23 March 2026, at the full 20 BTX each: 1,000,000 BTX, held in what the specification calls a genesis multisig "designated for network formation, including bridge seeding, relay infrastructure, and early market depth". The specification argues this differs from a premine because the coins were produced by open work and are managed as explicit on-chain holdings, and it says the operating policy "should be documented separately and precisely". That document does not exist as far as we can find. On-chain, most of that balance had been dispersed by late May. Call it what you like; it is 23 percent of everything issued so far, and you should weigh it.
4. **Privacy was promised and then removed.** The genesis message says "Shielded Transactions". Consensus stopped accepting new shielded deposits at block 125,000 (8 June 2026) after two Critical findings in the shielded proof system were closed structurally by sunsetting it rather than fixing it live. The pool was closed in both directions at block 199,300; whatever remained in it (about 21,000 BTX, 0.7 percent of supply, in July) is treated as burned. BTX today is a transparent chain. Dandelion++ relay privacy has been live since block 61,000, and it hides the originating node rather than the amounts. If privacy is why you are here, read that twice.
5. **Many nodes are not independently verifying the proof-of-work.** Exact replay needs a qualified GPU. CPU-only nodes follow signed attestations from GPU attestors under a pin they configure themselves. The project calls this topology temporary, publishes the transition plan, and the original operator has stopped committing to run signers. Today's census counts 31 of 60 public nodes as independent validators.
6. **No named third-party audit of the post-quantum stack, or of qID.** Internal and AI-assisted reviews, an unnamed external reviewer's NO-GO on the MatMul fork, formal verification of the shielded pool, and our own audits of the wallet and identity layers. All of that is diligence. None of it is assurance. Both projects say so in writing.
7. **No team page.** The GitHub organization lists no public members. Commits are authored by one account (Part 14). Community members refer to a pseudonymous lead architect. There is nobody to hold to a promise, and also nobody to enrich, no premine allocation to a founder, and no roadmap written to support a valuation. Bitcoin started the same way; weigh it either way you like.
8. **Bootstrap still runs through operator hosts, and more narrowly than the handover notice says.** Part 10.2. The three DNS seeds resolve to three addresses inside one hosting provider, and the two fixed seeds compiled in as the DNS-failure fallback are two of those same three. The source comment directly above them says, in the shipped release, that a two-entry subset of the same DNS set is not a fallback, and the checklist beside it asks for six to eight seeds across distinct networks. The fix is documented as an invitation, not a deliverable.

9. **Transactions are big.** About 3.7 KB per input. Bandwidth, archival storage and node requirements scale with it. This is the honest price of the entire idea, and BIP 360 would charge Bitcoin the same.
10. **The current core release ships checksums but is not signed**, and builds are not reproducible. Earlier releases were signed, so the signing stopped rather than never having existed. A checksum proves the file matches the upload, not that the binary matches the source.
11. **The layer 2 (EVX) is a design, not a product, and it is not a rollup.** Part 10.5. Its code is in a private repository, there is no public whitepaper, and btx.dev does not mention it. It runs its own consensus, so BTX proof of work is not what would secure it. The bridge pilot is small and its BTX leg ran on a local regtest chain rather than mainnet, so no mainnet BTX has been locked; the wrapped figure people quote comes from our own explorer, every transfer in it happened on one day in July 2026, and its reserve number is operator-set and stale. The wBTX contracts are labelled unaudited in seven separate files including the contracts themselves. Do not build on any of it this month.
12. **Ticker collision.** An unrelated 2017 coin has used the ticker BTX and is still listed. If you search for a price you will probably find that one.
13. **PQ Wallet has no SPV.** Part 7. It cannot lose your coins to a lying server, but it can be told you were paid when you were not.
14. **The model network is open for review, not released.** Part 9. On 14 September 0.34.7 was published as pull request 156, still open, still untagged, and the branch sets its own release flag to false. Anyone can read it now, which is a real improvement on a week ago. What has not changed is the evidence: the repository runs no continuous integration, so nothing in that pull request was checked by a machine, live multi-node relay has not been run, and the phrase monetary load isolation appears in none of the files it opened with.
15. **One person merges everything, and nobody has ever approved a pull request.** Part 10. Across all 105 pull requests the repository has ever had, the number of approving reviews by a human being is zero, and 41 of the 41 merges since June 2026 are his. Discussion is real and public, but there is no second signature on the way in.
16. **A compiled-in operator key can ship code to default nodes.** Part 10.2. Auto-update is on by default on mainnet, polls one operator-controlled URL every thirty minutes, and by default launches the verified installer. It is signed with a post-quantum key whose private half is stated to be offline, and it refuses anything not newer, so a stale endpoint is inert. It is still the one key in the tree with power over what nodes run, and the README's claim that no compiled-in signer key exists anywhere in the tree is true only of consensus.
17. **There are open consensus-level script findings, published by the project.** Pull request 137 records that ML-DSA signature non-canonicality and malleability, and CSFS signature replay across contracts sharing a hashlock and a key, are still being verified on a separate track, and that they touch the script interpreter. That is a cryptographic risk rather than a young-chain one, and it is the exact pattern a release campaign with many contributors would create, which Part 9 explains.



Item 2, quantified. Nakamoto's catch-up probability for an attacker holding 10, 20, 30 and 40 percent of the network's work, against the number of confirmations, with BTX's 90-second blocks on the upper axis. Dots mark where each curve first drops below one in a thousand: 5, 11, 24 and 89 confirmations. The BTX specification's own table lists 6, 11, 24 and 52; the first three agree with the formula (6 is one more than needed), the fourth does not, so a 40 percent attacker needs 89 confirmations, about 134 minutes, not 52. The formula is Satoshi Nakamoto's from the 2008 paper.

Only item 17 is a cryptographic weakness, and the project published it rather than us finding it. None of the rest is. Items 15 and 16 are governance and operations, which is where this chain's real risk now sits. Every one of them is a young-chain risk of the kind Bitcoin itself carried in 2010. The difference is that Bitcoin's remaining structural risk in 2026 is the one BTX was built to remove.

Part 13. Verify it yourself

Nothing in this article should require trusting us.

- **The rules.** github.com/btxchain/btx: `src/kernel/chainparams.cpp` for the constants (cap, subsidy, halving, 90-second target, every activation height); `doc/btx-pqc-spec.md` for the signature profile; `doc/btx-matmul-pow-spec.md` for the work; `doc/security/` for the audit closeouts; `doc/pq-benchmark-results.md` for the verification timings.
- **The chain.** btxscan.io for any block, address or transaction. Open any spend and look at the witness size. The JSON endpoints /api/network and /api/network-history are described in Part 5.4.
- **The specifications and papers.** btx.dev/docs/specs/post-quantum, btx.dev/docs/specs/matmul-pow, BTX-Specifications.pdf (the protocol paper), BTX-Context.pdf (the narrative), the institutional thesis and its "claims to avoid".
- **The standards.** FIPS 204 and 205 at csrc.nist.gov. postquantum.wiki for a cited plain-language reference on all of it.
- **Our audits.** Quoted in Part 4.6. The full reports will be published with the open-source release of the security-critical core.

- **Run a node.** easybtx.com/node for the one-click easyNode, or build `btxd` from source. A full node re-verifies every rule from genesis and is the only way to answer questions about the chain without trusting anybody's server, including ours.
- **The genesis message.** Pull block 0 from your node or the explorer and read the coinbase.

Part 14. People and repositories

BTX has no team page and names no founders. What can be checked:

- **github.com/numair (Numair Faraz)** is the account that authors the commits to `btxchain/btx`: 745 of the repository's 812 commits, across four git author addresses that all resolve to this one account. GitHub's own contributor graph undercounts that, and Part 10.1 says why and measures what the concentration does and does not put at risk. Part 10 measures what this concentration does and does not put at risk. Nobody has declared him the core developer and this dossier does not either; the commit list is public and takes ten seconds to read. According to the press archives cited in the BTX Handbook, his record predates crypto by two decades: an early Facebook music application covered by VentureBeat and Adweek in 2007, and marketing advisory work at Motorola before that.
- **"Chuck"** is the pseudonymous lead architect community members refer to. Nothing about him can be confirmed and nothing here speculates.
- **team@btx.dev** is the project's stated security contact.
- **github.com/btxchain/btx** is the reference node (MIT). **btx.dev** is the project's own site and documentation.
- **easyBTX**: the miner, the node app, this site and the research archive; **PQ Wallet** at pq-wallet.com; **qID** at qid.dev; the **bonuz** mobile wallet at bonuz.xyz. All share people with this article's publisher.
- Other ecosystem pieces we do not control: the mining pools, the `dexbtx/minebtx` pool and mining client, the MATADOR mining engine, and postquantum.wiki.

Appendix A. Sources

BTX project. btx.dev (overview, post-quantum specification, MatMul specification, institutional thesis, BTX-Specifications.pdf and BTX-Context.pdf dated 5 August 2026); github.com/btxchain/btx (README at v0.34.1, `doc/btx-pqc-spec.md`, `doc/btx-shielded-pool-guide.md`, `doc/security/README.md` and `current-status.md`, `doc/btx-security-audit-report-2026-06-06-reassessment.md`, `doc/btx-matmul-v4.2-external-audit-remediation.md`, `doc/btx-matmul-v4.4-independent-hardening-audit-2026-07-19.md`, `doc/pq-benchmark-results.md`, `src/libbitcoinpqc/README.md`); the GitHub API for contributors, tags and releases.

Standards and research. NIST FIPS 203, 204, 205 (August 2024); NIST IR 8547 draft; NSA CNSA 2.0; Roetteler, Naehrig, Svore, Lauter 2017 (arXiv:1706.06752); Webber and coauthors 2022 (arXiv:2108.12371); Gidney 2025 (arXiv:2505.15917); Aggarwal and coauthors 2017 (arXiv:1710.10377); Deloitte, "Quantum computers and the Bitcoin blockchain"; BIP 340, 341, 360;

Komargodski, Schen, Weinstein 2025 (arXiv:2504.09971); Fanti and coauthors, Dandelion++ (2018).

easyBTX. The BTX Handbook (second edition, 23 August 2026); the research archive (authenticity, useful proof-of-work, the MatMul v4.7 fork, the holder census, network status, why run a node); postquantum.wiki entries on BTX, qID, PQ Wallet, qID Connect, the quantum threat to ECDSA, taproot exposure and post-quantum blockchains; easybtx.com/stats, /api/network and /api/network-history on 11 and 12 September 2026; /api/btxprice (frozen 30 August 2026); our own explorer measurement of 30 May 2026.

Audits. The PQ Wallet client audit, the backend infrastructure audit and the hardening report of 25 July 2026; the qID security history, the audit brief of 11 August 2026 and the security maturity assessment of 17 August 2026; the qID Connect audit brief; the PQ Wallet changelog for v1.3.0. Held in our repositories, not yet public.

Appendix B. Glossary of technical terms

Alphabetical. Bitcoin terms are included where BTX reuses or changes them.

- **51 percent attack.** An attacker with a majority of the network's work can reorganize recent blocks and reverse their own recent payments. It cannot forge signatures, spend coins it does not own, or mint coins.
- **ACVP (Automated Cryptographic Validation Protocol).** NIST's program that publishes known-answer test vectors for standardized algorithms. qID pins its primitives to these vectors.
- **AEAD (authenticated encryption with associated data).** Encryption that also detects tampering; AES-GCM is the instance used in qID and PQ Wallet.
- **Argon2id.** A memory-hard password-hashing function. PQ Wallet derives seal keys from passphrases with it (64 MiB, 3 passes).
- **ASERT.** Absolutely scheduled exponentially rising targets: a difficulty algorithm that adjusts every block toward the target spacing using an exponential schedule. BTX uses the aserti3-2d variant from genesis, with a consensus difficulty floor since block 191,714.
- **Assumeutxo.** A node fast-start that loads a compiled-in snapshot of the unspent-output set and validates the history later. BTX's current pin is at height 201,500.
- **Attestation (MatMul).** A signed verdict from a GPU node that has exactly replayed a block's episode. CPU-only nodes may follow an M-of-N quorum of attestations under a pin they configure.
- **Bech32m.** The checksummed address encoding for witness versions 1 and up. BTX addresses use the `btx` human-readable part and witness version 2, giving `btx1z…`.
- **BIP 340 / 341 / 360.** Bitcoin proposals for Schnorr signatures, taproot, and pay-to-quantum-resistant-hash respectively. The first two are active on Bitcoin; the third is a draft.

- **Block weight and MWU.** A size measure that discounts witness data. BTX's block limit is 24 million weight units (24 MWU); Bitcoin's is 4 MWU.
- **Burn versus steal.** The unresolved Bitcoin governance question of what to do with quantum-exposed coins whose owners never migrate: freeze them by consensus or let an attacker take them.
- **C-002.** The BTX consensus upgrade at height 123,000 that switched SLH-DSA verification to FIPS-205 pure mode and hardened the shielded verifier.
- **Coinbase transaction.** The first transaction in a block, which creates the subsidy. On BTX it is the only path by which coins come into existence.
- **Consensus.** The rules every node enforces to decide which blocks and transactions are valid. Changing them requires a fork that participants choose to run.
- **Covenant.** A spending restriction on an output that constrains the transaction that spends it. CTV is BTX's covenant primitive.
- **CRQC (cryptographically relevant quantum computer).** A quantum computer large and stable enough to run Shor's algorithm against real key sizes. None exists as of 2026.
- **CSP (Content Security Policy).** A browser rule set that limits which scripts and connections a page may load. PQ Wallet and the qID servers run strict policies.
- **CTV (OP_CHECKTEMPLATEVERIFY).** An opcode that commits an output to the hash of the transaction template allowed to spend it. Live on BTX from genesis; used for vaults and payment trees.
- **CSFS (OP_CHECKSIGFROMSTACK).** An opcode that verifies a signature over arbitrary data supplied on the stack, enabling oracle-signed conditions. Live on BTX with ML-DSA and SLH-DSA signatures.
- **CLTV / CSV.** OP_CHECKLOCKTIMEVERIFY and OP_CHECKSEQUENCEVERIFY, Bitcoin's absolute and relative timelock opcodes, inherited by BTX.
- **Dandelion++.** A transaction relay protocol (BIP 156 style) that sends a new transaction along a random "stem" path before broadcasting it, so observers cannot easily tie a transaction to its originating IP. Live on BTX since block 61,000.
- **Descriptor.** A textual description of the scripts a wallet watches or signs for. BTX's default is `mr(<mldsa>, pk_slh(<slh>))`. needed to produce a valid proof, readable by anyone without holding coins.
- **Dilithium.** The pre-standard name of ML-DSA.
- **Domain separation.** Prefixing different kinds of messages with distinct tags before hashing, so a signature made for one purpose cannot be replayed for another. qID separates logins from spends this way.

- **ECDSA.** The elliptic-curve signature scheme Bitcoin has used since 2009. Not valid in BTX consensus.
- **ECDLP.** The elliptic-curve discrete logarithm problem, the hardness assumption under ECDSA and Schnorr, broken in polynomial time by Shor's algorithm.
- **Epoch A / B / C / D.** The four stages of BTX's MatMul v4.7 roadmap. Epoch A (live) uses exact replay as the authority; B adds a durable proof; C makes a succinct proof authoritative; D moves to the larger Profile 2 workload. of its episode on a qualified accelerator produces the identical digest.
- **Falcon / FN-DSA.** The third NIST post-quantum signature standard, compact but delicate to implement because of floating-point sampling. Opcodes reserved on BTX, not enabled.
- **FIPS 203 / 204 / 205.** The NIST standards for ML-KEM, ML-DSA and SLH-DSA, finalized 13 August 2024.
- **Freivalds' algorithm.** A probabilistic check that a matrix product is correct, by multiplying both sides by random vectors; quadratic instead of cubic cost. Used by MatMul v3.
- **Fuzzing.** Automated testing that feeds random or malformed inputs to code to find crashes and logic errors. BTX's post-quantum library ships fuzz targets.
- **GEMM.** General matrix multiply, the core operation of tensor-core hardware and of BTX's proof-of-work.
- **Genesis multisig.** The on-chain holding that received the 1,000,000 BTX produced during the 50,000-block bootstrap. The specification calls it a network-formation reserve.
- **Gini coefficient.** A 0-to-1 measure of inequality. BTX's holder census measured 0.88 in July 2026.
- **Grover's algorithm.** A quantum search algorithm with a quadratic speedup. It halves the effective bit strength of hash and symmetric primitives and is why proof-of-work is not the quantum problem.
- **Harvest now, decrypt later.** Recording encrypted data today to decrypt it once a quantum computer exists. On a public ledger, exposed keys are the signature-side equivalent.
- **Hash-based signature.** A signature scheme whose security depends only on a hash function. SLH-DSA is stateless; XMSS and LMS are stateful.
- **HD derivation (hierarchical deterministic).** Deriving many keys from one seed along a path. BTX uses purpose `87h` for P2MR.
- **KEM (key encapsulation mechanism).** A public-key method for agreeing a shared secret. ML-KEM is the post-quantum standard; KEM-DEM combines it with symmetric encryption.
- **Lattice-based cryptography.** Schemes whose security rests on problems over integer lattices (Module-LWE, Module-SIS). ML-DSA and ML-KEM are lattice schemes.

- **Logical versus physical qubit.** A logical qubit is an error-corrected unit built from hundreds to thousands of physical qubits. Attack estimates are quoted in both.
- **MAST / Merkle tree.** A tree of hashes whose root commits to all leaves; revealing one leaf plus its siblings proves membership. Taproot's script tree and BTX's P2MR are both Merklized script trees.
- **MatMul.** BTX's matrix-multiplication proof-of-work. v3 was a 512 by 512 multiply; v4.7 Profile 1 is a 4-round, 16-layer, 4,096-wide exact-integer episode.
- **Mersenne prime field.** Arithmetic modulo 2^31 – 1, chosen for MatMul v3 because it maps onto fast 32-bit integer operations.
- **ML-DSA.** Module-Lattice-Based Digital Signature Algorithm (FIPS 204). BTX uses ML-DSA-44, NIST category 2.
- **ML-KEM.** Module-Lattice-Based Key-Encapsulation Mechanism (FIPS 203), formerly Kyber. Used by qID (ML-KEM-768) and historically by BTX's shielded pool.
- **Module-LWE / Module-SIS.** Learning with errors and short integer solution over module lattices, the hardness problems under ML-DSA and ML-KEM.
- **Mosca's inequality.** If data shelf life plus migration time exceeds the time until a CRQC, protection has already failed.
- **NIST security categories 1 to 5.** Strength classes anchored to key search on AES-128, collision search on SHA-256, AES-192, SHA-384 and AES-256 respectively.
- **Nonce-bound seeds.** The MatMul hardening at height 125,000 that derives the work matrices from the nonce as well as the header, closing an amortized-mining finding.
- **Opcode.** A script instruction. BTX adds `OP_CHECKSIG_MLDSA`, `OP_CHECKSIG_SLHDSA`, their `CHECKSIGADD` variants, `OP_CHECKTEMPLATEVERIFY` and `OP_CHECKSIGFROMSTACK`. The elliptic-curve opcodes BTX inherited are unreachable, not deleted (Part 3.4).
- **P2MR (Pay-to-Merkle-Root).** BTX's only output type: witness version 2, a 32-byte Merkle root of post-quantum spend leaves, `btx1z` addresses.
- **P2PK / P2PKH / P2TR.** Bitcoin output types: pay-to-public-key (key exposed), pay-to-public-key-hash (key hidden until spend), pay-to-taproot (key exposed).
- **Passkey / WebAuthn / PRF.** The web standard for hardware-backed, origin-bound credentials (Touch ID, Face ID, security keys). The PRF extension derives a secret from the authenticator, which qID uses to seal the seed.
- **Premine.** Coins allocated before public mining. BTX has no protocol-level pre-allocation; see genesis multisig for the bootstrap production.
- **Profile 1 / Profile 2.** The live and the future MatMul v4.7 workloads. Profile 2 is eight rounds, 24 layers and about 16 times the work; not activated.

- **Q-Day.** The hypothetical date a CRQC first exists. Expert surveys concentrate probability in the 2030s and 2040s.
- **Reorganization (reorg).** Replacement of recent blocks by a competing chain with more work. BTX parks deep reorganizations and requires operator action beyond a fixed depth.
- **Schnorr signature.** Bitcoin's second signature scheme (BIP 340), used by taproot. Not valid in BTX consensus.
- **secp256k1.** The elliptic curve Bitcoin uses. Broken by a sufficiently large quantum computer.
- **Service challenge.** BTX RPCs that issue, solve, verify and redeem a fresh chain-bound MatMul work proof, so a service can gate expensive requests behind real computation (an AI-agent CAPTCHA).
- **SHA-256d.** Double SHA-256, Bitcoin's proof-of-work hash and BTX's final hash over the compressed MatMul transcript.
- **Shielded pool / SMILE v2.** BTX's original lattice-based confidential-transaction pool with ML-KEM note encryption, ring signatures, range proofs and a turnstile. New credits disabled at 125,000; closed in both directions at 199,300.
- **Shor's algorithm.** The quantum algorithm that factors integers and computes discrete logarithms in polynomial time, breaking RSA, ECDSA and Schnorr.
- **Sighash.** The digest a signature commits to. BTX's P2MR sighash follows BIP 341's structure with its own epoch byte and commits to input amounts.
- **Sigop weight / validation weight.** The consensus cost charged per signature check. SLH-DSA checks are charged ten times ML-DSA checks; blocks are capped at 480,000 sigop cost.
- **SLH-DSA.** Stateless Hash-Based Digital Signature Algorithm (FIPS 205), formerly SPHINCS+. BTX uses SLH-DSA-SHAKE-128s, NIST category 1.
- **Soft fork / hard fork.** A rule tightening that old nodes still accept, versus a rule change that old nodes reject. BTX's Falcon slot is designed for soft-fork activation; MatMul v4.7 was a hard fork at a fixed height.
- **SPV (simplified payment verification).** Checking a payment against block headers and a Merkle proof instead of trusting a server. PQ Wallet does not have it yet.
- **Succinct proof.** A short certificate that lets a verifier check a computation without redoing it. Designed for later MatMul epochs; no activation height.
- **Taproot.** Bitcoin's witness-version-1 output type with a tweaked key and a script tree. Its key sits in the output, so taproot coins are quantum-exposed at rest.
- **Tensor cores.** GPU units specialized for dense matrix multiply, the hardware BTX's work function targets.
- **TOCTOU (time of check to time of use).** A bug class where a value is validated and then a different value is used. One was found and closed in a qID backend probe.

- **Trusted mirror.** A BTX node mode that validates everything except MatMul locally and accepts an M-of-N quorum of attestations for the work. Not an independently validating full node.
- **Turnstile.** The shielded-pool accounting rule that the value leaving the pool can never exceed the value that entered it.
- **UTXO.** Unspent transaction output, the unit of Bitcoin-style accounting. BTX keeps the model unchanged.
- **Witness version.** The version byte of a segregated-witness output. Bitcoin uses 0 (P2WPKH, P2WSH) and 1 (taproot); BTX uses 2 (P2MR).
- **X25519.** A classical elliptic-curve key agreement, combined with ML-KEM in qID's hybrid encryption so that the construction is no weaker than either part.
- **Assurance contract.** A pledge that only binds if total pledges reach a threshold, which is what makes threshold crowdfunding work for public goods. BTX's release-financing design is not one; the published code shows it, rather than a description (Part 9.5).
- **`btx://` (BTX resource URI).** A compact address for one exact artifact, carrying a version, a kind and a 48-byte SHA-384 digest of the bytes it names. It is not a payment instruction and opening one does not spend anything.
- **`btx-modeld`.** The separate daemon that fetches and serves model files. It is not the money daemon, it runs as its own process, and the node refuses model calls when it is not running.
- **ML-KEM-768.** The NIST post-quantum key encapsulation standard, FIPS 203, used to agree a shared secret rather than to sign anything. BTX consensus does not use it. The model network's transport pins it, and refuses the hybrid modes that pair it with a classical exchange.
- **Fair exchange.** Trading two items so that neither party can take one without giving the other. Pagnia and Gärtner showed it is impossible in general without a trusted third party, which bounds what any hashlock can promise.
- **Free-rider problem.** Once a good is public, nobody needs to pay for it, so rational contributors wait. It is the central difficulty in funding public goods, and publishing a decryption key makes it sharper rather than softer.
- **Hashlock and preimage.** A script clause that pays only to whoever reveals a value whose hash matches a commitment. The revealed value is the preimage, and on BTX it lands in the public claim witness.
- **HTLC (hash time-locked contract).** An output spendable either by revealing a preimage or, after a timeout, by refund. BTX's shipped form is `mr(htlc_sha256(...), refund(...))`. Loading a pickle file can execute arbitrary code, which is why refusing it is the correct decision for any network that distributes weights.
- **Street Performer Protocol.** Kelsey and Schneier, 1999: the public escrows donations that are released to an author when the work enters the public domain. The ancestor of the release-financing design in Part 9.

- **XMSS / LMS.** Stateful hash-based signature schemes; each one-time key may sign once. Not used by BTX, which chose the stateless SLH-DSA.

For education only. Not financial advice. BTX has no established public market, mining may earn nothing, and you are responsible for your own decisions.

Frequently asked questions

If the lead developer stopped, would BTX fail?

Not in the layer that makes it money, and yes in the layer that makes it software. There is no compiled-in signer key in consensus, so a node accepts a block because it replayed the work itself, and your coins sit behind your own keys. In August the network produced and accepted blocks past 199,328 while both operator-run nodes sat at 199,300, though it did so during a stall. What does depend on him is everything around that: he wrote 745 of the repository's 812 commits and merged 41 of the 41 merges since June 2026, no human has ever approved a pull request, and new nodes bootstrap through three DNS names that resolve into a single hosting provider. Part 10 measures all of it.

Does BTX distribute AI models now?

No. On 13 September 2026 the lead developer described a coming release, 0.34.7, adding a decentralized network for distributing open-weight models and financing their release. On 14 September he published it for review as pull request 156, which was 240 files and 33,367 added lines when it opened and has grown since. It is still not a release: no tag, unmerged, and the branch sets its own release flag to false. One piece had already shipped the morning before in v0.34.6, the hash-locked escrow the financing uses, carrying an alias named model_htlc_sha256. Part 9 separates what is tagged from what is merely readable, and lists four things the mechanism still cannot do.

Is BTX actually post-quantum, or does it just say so?

It is post-quantum only, not post-quantum available. Consensus rejects any block containing a non-OP_RETURN output that is not a witness-version-2 P2MR output, so no classical output type can exist, and the elliptic-curve code BTX inherited from Bitcoin Knots is unreachable rather than deleted. Every coin sits behind a P2MR output whose spend leaves verify ML-DSA-44 (FIPS 204) or SLH-DSA-SHAKE-128s (FIPS 205) signatures. You can confirm this from the reference source and from the witness of any transaction on the explorer.

Who has audited BTX's post-quantum cryptography?

Nobody outside the project has signed off on it, and the project says so. The core repository carries its own dated audit archive (a March 2026 source audit, a June reassessment, an external NO-GO audit of the MatMul v4 fork with a remediation map, and a formal-verification suite for the shielded pool). Our own wallet, identity and backend layers went through adversarial execution-driven audits in June, July and August 2026. All of that is diligence, not independent assurance.

What are ML-DSA and SLH-DSA, and why does BTX use two schemes?

ML-DSA (formerly Dilithium) is a lattice-based signature standardized in FIPS 204; SLH-DSA (formerly SPHINCS+) is a hash-based signature standardized in FIPS 205. BTX puts ML-DSA-44 in the primary spend leaf and SLH-DSA-128s in a backup leaf of every output. The two rest on independent hardness assumptions, so a break in lattice cryptography would leave every coin spendable through its hash-based path.

When is the first BTX halving?

At block 525,000. As of 12 September 2026 the chain is at block 217,689, so 307,311 blocks remain. At the 90-second target that is about 320 days, which puts the first halving around late July 2027. The subsidy drops from 20 to 10 BTX, and half of all BTX will exist at that point.

Was there a premine?

Not at the protocol level: coins are created only by mining. But blocks 1 to 50,000 were mined at a 250-millisecond cadence in about six and a half hours on 23 March 2026, at 20 BTX each, and the resulting 1,000,000 BTX went to what the specification calls a genesis multisig for network formation. That is about 23 percent of everything issued so far. This dossier states it plainly so you can weigh it.

Does BTX have private transactions?

Not any more. The shielded pool named in the genesis message stopped accepting new deposits at block 125,000 (8 June 2026) and was closed in both directions at block 199,300. BTX today is a transparent chain. Dandelion++ relay privacy has been live since block 61,000, and it hides the originating node, not the transaction.

Is BTX mining AI inference?

No. The work is transformer-shaped exact-integer matrix arithmetic on the same accelerator class that runs inference, but the inputs are seeded from the chain and only a digest is kept, so no outside party receives a result. What is real today is the hardware fleet the mining funds and a shipping admission-control primitive that uses fresh work proofs as an AI-agent CAPTCHA.

Who wrote this, and how independent is it?

It was researched and written by Claude Fable 5.1, an AI model, from public sources and from the audit records in our own repositories, under our direction and reviewed before publication. It is independent of the BTX core team, which easyBTX has no part in. It is not independent of easyBTX: we build software for BTX, hold BTX, and take a fee from mining, and the text says so.

Published by easyBTX, which builds software for BTX and is not the BTX core team. This paper is educational research, not financial advice, and it is not an official BTX document. Read the live version and check the sources at easybtx.com/research/btx-post-quantum-dossier.