

ShieldBit

A Peer to Peer Currency Designed for Post Quantum Signatures

Vipul Tomar (findance)

Whitepaper version 1.0 | 2 October 2026

shieldbit.org | [Source code](#)

Abstract

Money kept for years should not depend on a signature scheme that becomes unsafe before its owner next needs it. ShieldBit starts with that concern. It proposes an independent proof-of-work currency, SBT, with ML-DSA-65 as its intended transaction signature scheme, a capped issuance schedule, and payment channels built into the protocol.

The design keeps familiar foundations: an unspent-output ledger, publicly verifiable rules, ten-minute target blocks, and mining rewards that halve every 210,000 blocks. Its distinctive contribution is the combination of versioned post-quantum signatures and native channel settlement. State anchors and escalating challenge bounties are intended to make dishonest channel closure easier to detect and less attractive.

ShieldBit also provides a foundation for lighter wallets, optional recipient privacy, and conditional payments. Each serves a practical purpose: fewer repeated downloads, less routine address reuse, and a way to exchange value without handing custody to an intermediary. The project is open source under Apache-2.0, with no pre-mine in its intended launch policy.

This paper describes the design and reference implementation as they stand in October 2026. The public website and automated test workflow are live; mainnet is not. Cryptographic integration, consensus review, and public testnet validation remain part of the path to production. The opportunity is a payment system whose long-term security choices are explicit from the beginning.

Why ShieldBit

ShieldBit gives contributors a focused problem to work on: preserve the simplicity of peer-to-peer money while preparing its signing rules for a different cryptographic future. The code is available for inspection today, and the design leaves room for measured upgrades rather than requiring every assumption to remain fixed forever.

1 Purpose and Monetary Rules

A currency people can understand

A payment should be understandable to the person making it. A node should be able to explain why it accepted it. ShieldBit follows that principle by separating monetary policy, transaction validation, and cryptographic providers. The intent is to make the rules easier to inspect, test, and maintain as the system grows.

SBT is the currency unit. One SBT contains 100,000,000 quanta. Amounts are integer values; no floating-point arithmetic is needed to decide how much a transaction spends or creates. The proposed supply ceiling is 21,000,000 SBT, with an initial subsidy of 50 SBT per block and halvings every 210,000 blocks.

Parameter	Reference setting
Currency ticker	SBT
Supply ceiling	21,000,000 SBT
Smallest unit	1 quanta = 0.00000001 SBT
Initial block subsidy	50 SBT
Subsidy halving interval	210,000 blocks
Target block interval	600 seconds
Maximum serialized block size	6 MiB
Difficulty interval	2,016 blocks
Coinbase maturity	100 blocks
Minimum transaction fee	250 quanta
Dust threshold	546 quanta

The subsidy schedule is defined in quanta as $5,000,000,000$ shifted right by $\text{floor}(\text{height} / 210,000)$. Integer rounding eventually brings the subsidy to zero. A block producer may collect the permitted subsidy and transaction fees; the supply ceiling is an upper bound, not a promise of market value.

The 250-quanta minimum is a validation and relay rule in the reference design. It does not guarantee that every payment will be confirmed for that amount when block space is scarce. Fees, propagation, and channel liquidity need measurement under realistic workloads.

The launch policy provides no pre-mine or separate founder allocation. Canonical genesis parameters and launch procedures still need to be finalized and reviewed. A published issuance policy gives participants a common rule to verify rather than a discretionary promise to trust.

2 Post Quantum Signatures and Cryptographic Agility

Planning for keys that remain useful

A sufficiently capable quantum computer would threaten classical signature systems based on discrete logarithms. ShieldBit selects ML-DSA-65, the module-lattice signature scheme standardized in NIST FIPS 204, as its intended spending authorization mechanism. NIST describes ML-DSA as believed secure against adversaries with a large-scale quantum computer. This is a security assumption supported by standardization and public analysis, not immunity from future discoveries. [1]

The reference design assigns version 0x01 to ML-DSA-65. Public keys and signatures carry a version byte so validation can select the corresponding provider. Version 0x02 is reserved for SLH-DSA, the stateless hash-based family standardized in FIPS 205; its provider is currently a placeholder. Reserving a version makes room for another approach without suggesting it is already active. [2]

ML-DSA-65 uses a 1,952-byte public key and a 3,309-byte signature before ShieldBit's version prefix. Larger signatures are a real engineering cost. ShieldBit responds with a 6 MiB block limit, compact relay, and a channel model that can move repeated balance updates off chain. Throughput and latency must still be established with benchmarks. [7]

SHA3-256 provides the main family of internal commitments. Transaction identifiers use double SHA3-256. Block headers use double SHA-256 for proof of work. The current address identifier is the first 20 bytes of SHA3-256 applied to the public-key material, rather than a full 256-bit digest. These choices have different security margins and should be evaluated separately. [3, 8]

The provider boundary

The implementation delegates key generation, signing, and verification to a cryptographic library. It is designed to fail when the selected provider is unavailable, rather than substitute a different scheme. This boundary keeps custom signature mathematics out of the currency implementation.

Before release, the adapter must be aligned with the library's supported ML-DSA-65 classes and key-loading methods, and verified with known-answer and round-trip tests. The current library documents `MLDSA65PrivateKey` and `from_seed_bytes`; the repository adapter still uses different names. A passing general test suite does not establish that this integration is ready. [7, 8]

The positive commitment is precise: post-quantum spending signatures are a first-class design choice, with a clear place to validate and upgrade them.

3 Transactions and Proof of Work

Independent verification

ShieldBit uses an unspent transaction output model. A transfer names earlier outputs, supplies authorization, and creates new outputs. Validators check the referenced funds, permitted amounts, signatures, and fees before changing ledger state. This keeps ownership grounded in explicit records rather than balances maintained by a central service.

Transactions have a version, a type, inputs, outputs, separate lock-height and lock-time fields, a replacement flag, and type-specific data. Amount bounds and distinct input references are part of validation. Separating height from wall-clock time makes a payment's conditions easier to state and inspect.

The signing digest is computed from a serialization with input signature fields empty. The transaction identifier, however, is currently computed from the full serialization, including signatures. These are different commitments. Excluding signatures from the signing digest avoids a circular signing dependency; it does not by itself establish a signature-independent transaction identifier or eliminate transaction malleability. [8]

Block-reward transactions include their height to distinguish rewards across blocks. Coinbase maturity separates newly issued value from immediately spendable funds. These are useful foundations for testing supply accounting, duplicate-output handling, and rollback behavior.

Ordering and accumulated work

Proof of work establishes the ordering cost of the ledger. The reference header is a canonical 152-byte record containing version, previous hash, transaction Merkle root, filter root, 64-bit timestamp, compact target, full target, and 64-bit nonce. Its double-SHA-256 digest must satisfy the applicable target. Chain selection is intended to follow accumulated validated work. [4, 8]

Difficulty parameters use a 2,016-block interval, with adjustment bounded to one quarter through four times the expected duration. Median-time-past constrains timestamps. Exact retarget arithmetic, header serialization, mining, and validation need agreement across implementations before the consensus rules are frozen.

Double-SHA-256 is a useful foundation for Bitcoin-oriented mining tooling. It does not alone enable Bitcoin merge mining. Auxiliary proof verification, parent-chain commitments, and pool integration remain development work; this paper makes no claim that Bitcoin hash power already secures ShieldBit.

Finality is probabilistic. More confirmations increase the work required to replace a payment, while reorganizations, concentrated mining, and network partitions remain relevant. Signature security protects spending authorization; it does not replace consensus security.

4 Native Payment Channels

Repeated payments without repeated settlement

Many payments happen between people or services that already expect to transact again. Recording every small balance change on the base chain can be wasteful. ShieldBit's channel design locks capacity on chain, exchanges signed balance states off chain, and returns to the ledger for settlement or a dispute.

Channels are represented by native transaction types for opening, anchoring, cooperative closure, force closure, and penalty submission. This gives developers a defined settlement vocabulary to build around. Off-chain updates can be quick when both parties cooperate and sufficient capacity is available; they are not a promise of unlimited routing or guaranteed liquidity.

Each commitment state has an increasing state number. An on-chain anchor records a state reference. The current validator rejects a force-close state number at or below the latest anchor, requiring the submitted state to be newer. That exact equality rule needs review alongside anchor publication and closure behavior so an honest participant always retains a usable exit. [8]

A reason to watch for dishonest closure

The dispute model gives a participant or an authorized watcher time to present revocation evidence. Its challenge bounty increases across three phases, paid from the closer's claimed allocation. The design turns monitoring into an economically motivated activity while bounding the settlement delay.

Phase	Elapsed blocks after force closure	Bounty rate
First	0 through 143	5 percent
Second	144 through 287	15 percent
Third	288 through 431	20 percent
Finalization	432 or more	Challenge window expired

At the ten-minute target, 432 blocks correspond to roughly three days. Actual elapsed time depends on block production. A watcher must hold the necessary evidence and get a valid challenge included before the deadline; an incentive cannot guarantee either condition.

For a closer claiming 100 SBT, the phase rates correspond to 5, 15, and 20 SBT. This illustrates the bounty calculation, not a guaranteed award or a measured attack deterrent. An honest unchallenged close is intended to recover withheld value at finalization.

State anchors, revocation evidence, and bounded challenge phases offer a promising combination. Their security depends on complete authorization checks, correct balance accounting, and tested settlement paths. Channel review and adversarial testnet exercises will establish how well those pieces work together.

5 A Network Built for Practical Verification

Send what the recipient needs

Post-quantum signatures make efficient propagation valuable. ShieldBit includes compact block relay: peers send a header, selected prefilled transactions, and short identifiers, allowing a receiving node to reconstruct a block from transactions already in its mempool. Missing transactions can then be requested separately.

The benefit depends on mempool overlap, transaction size, and network conditions. Short identifiers are retrieval hints, not a substitute for checking full transaction data. The project does not claim a fixed percentage bandwidth reduction or a measured transaction rate at this stage.

Compact Golomb-coded filters provide another useful building block. A wallet can test downloaded filters locally for relevant recipient commitments and request candidate blocks. This follows the broad approach of BIP 158 while using ShieldBit's own parameters and commitment conventions. It is not a claim of wire compatibility with Bitcoin filters. [6]

A light wallet may avoid downloading every full block, but it also takes on additional trust and availability assumptions. False positives are expected in probabilistic filters. Header validation, honest data availability, and verification of requested transactions still matter. Requests for matching blocks can reveal information about wallet activity.

Maintainable node software

The reference implementation separates consensus, transaction structures, cryptographic providers, channels, network messages, wallet handling, and storage. Its Python code and SQLite persistence make the state transitions accessible to contributors. The aim is to keep a change in one area reviewable without hiding the rules behind a large application framework.

Peer messages use explicit framing, bounded payload lengths, checksums, and version negotiation. The reference network port is 9333. Seed discovery, peer diversity, resource limits, and recovery from disconnected or malicious peers need operational testing before a public network can depend on them.

Storage includes indexes for blocks, outputs, filters, and channel state. Reliable operation also requires persistence to agree with validation during crashes and reorganizations. Pruning and mobile-wallet policies should be specified and measured before being advertised as production capabilities.

The practical strength is a set of understandable components for efficient verification. The next step is to measure them under load and confirm their behavior when peers fail, blocks arrive out of order, or the chain reorganizes.

6 Privacy Recovery and Conditional Payments

A useful choice for recipients

Transparent payments remain the default. Addresses use Bech32m encoding with the sb human-readable prefix, producing addresses beginning sb1. The checksum helps detect entry errors; it does not prove that a recipient is trustworthy or that a payment is private. [5]

The optional stealth-address module derives one-time recipient commitments from scanning and spending material. It is intended to reduce routine address reuse. Its current scan-key agreement uses X25519, a classical Diffie-Hellman construction; the spending design uses ML-DSA. Post-quantum spending authorization therefore does not imply post-quantum scanning privacy.

Amounts and ledger activity are not hidden merely by using a one-time recipient. Network metadata and transaction patterns may also expose relationships. A production privacy mode needs a reviewed spending path and a post-quantum key-establishment design, as well as a clear explanation of what it protects.

Recovery without one fragile backup

The wallet includes encrypted key storage using AES-256-GCM and threshold backup helpers based on Shamir secret sharing. A k-of-n arrangement permits recovery from any k valid shares, allowing an owner to distribute backup material across separate locations.

For example, a three-of-five backup can tolerate the loss of two shares. Possession of three valid shares also permits recovery, so their storage needs to reflect that authority. Secret sharing is a recovery mechanism, not threshold transaction signing. Password strength, authenticated share formats, secure randomness, and reliable restoration remain part of wallet review.

Hash and time locks

An HTLC conditions payment on a secret preimage and an expiry height. Revealing the matching preimage enables a claim; expiry permits a refund path. This supports a foundation for routed payments and cross-chain exchange without assigning custody to a swap operator.

The current payment-hash helper uses SHA3-256. A swap requires both chains to enforce compatible hash functions and claim conditions, with timeout ordering that leaves enough room for confirmations and refunds. Bitcoin's usual SHA-256 HTLC construction is not automatically compatible with ShieldBit's current helper.

Cross-chain adapters, routing, and tested reorganization handling remain work to complete. The benefit is the conditional-payment primitive and an open integration path, rather than a claim of universal atomic swaps already operating.

7 Open Development and the Path to Launch

A public foundation

ShieldBit's reference implementation is available on GitHub under Apache-2.0. The project publishes its code, network parameters, tests, and this paper so contributors can examine the design directly. The website is live at shieldbit.org, with updates deployed from GitHub Actions.

The repository's 127-test workflow has completed successfully in GitHub Actions. That is useful evidence of repeatable checks over the covered behavior. It is not an independent audit, a complete test of all advertised features, or proof that live signing and network operation are ready.

Development follows the founder-led governance model documented by Vipul Tomar, known as findance. During his lifetime he approves protocol changes and repository merges. The stated succession policy is community rough consensus after his death. This is development governance; a production network's participants would still choose which software and consensus rules they run.

Evidence before activation

The next release work is to align the ML-DSA provider with supported library interfaces, validate key encodings and signature vectors, and exercise complete wallet and channel flows. Consensus review must then confirm exact header hashing, issuance bounds, reorganization behavior, anchor rules, and settlement authorization.

A public testnet should follow with separate network identity, published genesis parameters, seed nodes, a faucet, and an issue-reporting process. Its purpose is to observe propagation, recovery, channel disputes, and resource use in conditions that unit tests cannot reproduce. Independent cryptographic and application review, signed releases, and operator guidance precede mainnet activation.

Merge-mining integration and post-quantum stealth scanning are further development targets. Neither is represented here as a deployed capability. Launch dates should follow the results of review and testing rather than substitute for them.

What ShieldBit contributes

ShieldBit puts a long-term signature choice, predictable monetary rules, and native channel settlement into one inspectable design. Its strongest promise is the opportunity to build and verify those pieces together while the system is still young.

The project welcomes practical scrutiny: a reproducible failure, a clearer rule, a better test vector, or a measured improvement in relay is a meaningful contribution. Questions and security reports can be directed to support@shieldbit.org. Source code and implementation references accompany the paper for readers who want to check the details themselves.

8 References and Implementation Notes

Standards and prior work

- [1] NIST. FIPS 204, Module-Lattice-Based Digital Signature Standard. 2024. [Standard and publication record](#).
- [2] NIST. FIPS 205, Stateless Hash-Based Digital Signature Standard. 2024. [Standard and publication record](#).
- [3] NIST. FIPS 202, SHA-3 Standard. 2015. [Standard and publication record](#).
- [4] Satoshi Nakamoto. Bitcoin A Peer-to-Peer Electronic Cash System. 2008. [Original paper](#).
- [5] Pieter Wuille. BIP 350, Bech32m. [Specification](#).
- [6] Alex Akselrod and Jim Posen. BIP 158, Compact Block Filters for Light Clients. [Specification](#).
- [7] Python Cryptographic Authority. ML-DSA signing documentation. Consulted 2 October 2026. [Provider interfaces and parameter sizes](#).

Reference implementation

- [8] Vipul Tomar (findance). ShieldBit reference implementation, version 0.1.0. Technical review baseline dcc5781, 2 October 2026. [Immutable source snapshot](#).

Monetary parameters are in `shieldbit/consensus/params.py`. Header encoding and proof-of-work logic are in `shieldbit/core/block.py` and `shieldbit/consensus/pow.py`. Transaction commitments are in `shieldbit/core/transaction.py`. Signature versions and provider integration are in `shieldbit/crypto/keys.py`.

Channel anchors, bounties, and conditional payments are in `shieldbit/channels/`. Validation and ledger application are in `shieldbit/consensus/validation.py` and `shieldbit/core/blockchain.py`. Address commitments and stealth scanning are in `shieldbit/crypto/ hashing.py`, `address.py`, and `stealth.py`. Compact filters and relay are in `shieldbit/core/filters.py` and `shieldbit/network/compact_relay.py`.

This version records both implemented structures and intended protocol behavior. It does not freeze consensus or report an audited production deployment. Where documentation and executable code differ, the differences must be resolved through review before a release becomes canonical. Performance figures, active users, mining partners, and exchange support are not claimed without corresponding evidence.

Copyright 2026 Vipul Tomar (findance). Project source is licensed under Apache-2.0.