

PIVX decentralised voting audit

Implementation, SHIELD privacy, proposal comparison and delivery plan

Prepared 20 October 2026 | Evidence-backed engineering assessment

Recommended foundation: equal voting weight per eligible PIV, implemented on the PIVX chain, with private eligibility proofs and consensus-enforced settlement. This is a new design derived from VFA principles, not an endorsement of VFA unchanged.

None of the four 2017 documents is a complete specification for the stated goal. Each leaves critical privacy, availability, identity or enforcement questions unresolved. Adding more eligible voters while retaining a privileged final approval or enforcement switch would not meet the goal. [P1] [P2] [P3] [P4] [P5]

Proposal	Useful contribution	Main obstacle for this goal
Tango	Simple reward-derived entitlement	Passive SHIELD holders are excluded; votes follow public reward addresses.
Weighted Voting	Broad participation and explicit thresholds	Unequal holder classes, difficult private classification, constitutional rules needing interpretation.
Votes-For-All	Equal weight per PIV; snapshot concept	The proposed second chain and private ticket issuance are unspecified.
Pivmetheus	Attention to minority protection and status quo	Account-based approximations, non-linear weights and long-term tuning need validation.

What was audited

Source-level governance, Sapling, staking, state, control-switch and treasury paths were inspected at Core bd8a672 (9 October 2026), compared with release v5.6.1 / af60f19642, and traced into the pinned Rust dependency e7662b23. CDG revision 4806f817 (5 November 2017) was reviewed. Runtime testing uses the official v5.6.1 Linux release and release-matched tests.

This is a comprehensive feasibility and architecture review for governance, supported by selected execution tests. It is not an exhaustive line-by-line security certification of every PIVX component, a formal cryptographic proof, or a tested implementation of a new CDG protocol. CDG contains proposal documents rather than an executable implementation.

Planning decision

Budget 12-18 months for a reviewed single-chain design with private eligibility and publicly countable anonymous ballots, subject to a successful cryptographic prototype. If permanently secret ballots and no special decryption committee are mandatory, use a research gate and a provisional 18-30+ month horizon. Do not commit to mainnet before the privacy and liveness design is demonstrated.

02 / Define fully decentralised voting

For this audit, decentralisation is a set of testable properties rather than a claim that a blockchain has no trust assumptions. PIVX still relies on its consensus participants, network connectivity and cryptographic assumptions. The goal is to avoid adding privileged governance authorities.

Required property	Acceptance condition
Open eligibility	Any qualifying self-custodial holder can participate, including SHIELD holders, without operator approval.
Independent verification	A fresh validating node can derive eligibility, accepted ballots, tally and payouts from canonical chain data and proofs.
No outcome administrator	No website owner, coordinator, spork key or masternode ratification group can choose the result.
Available participation	Anyone can submit or relay a valid ballot. Full-node operation remains a fallback if a wallet server is unavailable.
Deterministic execution	Valid results execute under fixed rules and treasury caps; operator discretion is not a final step.
Private eligibility	Participation does not expose spending keys, viewing keys, note ownership, precise holdings or payment nullifiers.
Explicit residual assumptions	Consensus censorship, custody, lending, ballot buying, endpoints and any decryption threshold are documented.

Decentralisation is different from equal influence

One PIV equals one vote is invariant under splitting the same eligible balance across accounts, but wealthy holders retain more influence. It is a stake-holder governance model, not one-person-one-vote. An exchange holding customer keys can exercise the corresponding cryptographic authority unless users withdraw or the custody model changes.

PIVX addresses and private notes do not identify unique humans. Capping votes per address, account or wallet does not establish one-person-one-vote. Adding human uniqueness requires another Sybil-resistance assumption, identity/attestation mechanism or resource model; none of these proposals supplies a completed solution.

A holder vote can authorise treasury spending or an allowlisted parameter transition under already adopted software. It cannot force independent users to install arbitrary new software or prevent a community choosing a different fork.

The source also has spork-gated quorum ChainLocks verification and enforcement. Any enabled base-chain finality mechanism remains an inherited assumption. This review did not establish its live mainnet status. Removing every special role from the currency itself would require additional base-consensus work beyond the governance estimate. [\[C30\]](#)

03 / Execution: environment and observations

Actual execution: official PIVX v5.6.1 daemon and test binary on isolated regtest, with release-matched source tests. No real coins, mainnet payments or new voting protocol were involved.

Item	Captured result
Source inspected	Core bd8a672d1d959db7cc6157e40c89dc1829d87abc
Runtime/source match	v5.6.1.0-af60f19642 / af60f19642c6356061049df253e8036e81879f4e
Release integrity	SHA-256 matches the official release manifest. The PGP signature was not independently verified. [T1]
Source build attempt	./autogen.sh stopped: autoconf missing. Boost, libevent, libsodium and Berkeley DB development headers were also unavailable. No successful source build is claimed.
SHIELD smoke exercise	Activated Sapling at regtest height 201; shielded 9 PIV; transferred 2 PIV to a second wallet. Recipient balance: 2.00000000 PIV.
Wallet-local balance query	The observer wallet reported zero for a shielded address it did not own. This illustrates local wallet knowledge, not a quantitative privacy proof.
Existing eligibility	This is not a masternode. 'local' option disabled.
Network controls	Default spork values recorded only on isolated regtest. These are not current mainnet settings.

The release test wrapper supplies the repository Sapling parameters explicitly and disables discovery/automatic outbound services. Test nodes communicate locally. The evidence package contains version pins, commands, selected logs, XML unit reports, observations and model calculations.

CDG itself has no executable voting implementation to run. This audit ran existing PIVX software to validate relevant behaviour; it did not implement any of the four proposals or certify a future private voting construction.

04 / Execution: selected test results

Focused unit coverage: 40 passing cases

Suite/filter	Passed	Failed
budget_tests	8	0
validation_tests/test_simple_shielded_invalid	1	0
coins_tests	13	0
network_upgrades_tests	4	0
sapling_note_tests	2	0
sapling_key_tests	2	0
sapling_keystore_tests	3	0
sapling_merkletree_tests	4	0
noteencryption_tests	2	0
zip32_tests	1	0

Release-matched multi-node functional tests

Test	Result	Duration
rpc_budget.py	PASSED	66 s
sapling_wallet_nullifiers.py	PASSED	159 s
sapling_malleable_sigs.py	PASSED	47 s
sapling_wallet_anchorfork.py	PASSED	388 s
tiertwo_governance_reorg.py	PASSED	394 s
tiertwo_governance_invalid_budget.py	PASSED	256 s
mining_pos_coldStaking.py	PASSED	149 s

These selected tests exercise budget APIs, nullifiers, shielded signature handling, anchor/reorg behaviour, governance reorganisations, invalid budgets and cold staking. [T2] They are not a complete PIVX suite or a test of a new private voting protocol.

An initial combined unit run aborted one fixture because its default Sapling parameter location was missing. That incomplete run is retained in the evidence and is excluded from the 40 passing-case count. Parameter-dependent SHIELD operation was subsequently exercised using the explicit repository parameter directory.

05 / Current Core: decisive findings

Finding	Evidence	Required change
Public ballots and masternode eligibility	Ballots serialize collateral input, choice and timestamp. Eligibility checks masternode keys. [C1] [C2] [C3]	Introduce a new proof-authenticated ballot type and entitlement model.
Second approval stage	Masternodes separately sign finalized budgets, ordinarily checking winning proposals in auto mode. [C5] [C6]	Replace political ratification with deterministic, verifiable settlement.
Separately synchronized state	Budgets persist in budget.dat; payee checks return early while tier-two state is not synchronized. [C7] [C9]	Define replayable governance state and proof/data availability from chain history.
Privileged control path	Sporks are authenticated by configured public keys. Budget and Sapling maintenance paths consult them. [C21] [C22] [C23]	Remove or sunset their authority over the new governance rules at an agreed upgrade.
SHIELD is a note system	Wallets decrypt notes locally; public chain data contains commitments, ciphertexts and spend nullifiers. [C11] [C14] [C18]	Prove entitlement without constructing a public address-balance registry.
SHIELD staking rejected	Sapling data is rejected in coin stake and coinbase transactions. [C10]	Tango cannot automatically include entirely shielded passive holdings.

These are architecture observations and requirements for the requested design. They are not assertions that a specific exploit has been demonstrated against the live PIVX network. No mainnet spork-state survey or ownership-distribution measurement was performed.

Mainnet v6 activation is unset in the inspected source, although deterministic masternode paths are present. Existing development code should not be described as an already activated mainnet feature. [\[C23\]](#) [\[C24\]](#)

The current passing check uses `net Yes greater than floor(enabled masternodes / 10)`. Final budget eligibility has separate thresholds, including a 5% support check. These are budget rules, not automatic software-upgrade approval rules. [\[C4\]](#) [\[C6\]](#)

06 / SHIELD privacy and entitlement

Membership is not an unspent-balance proof

Sapling commitments remain in the note tree after spending. A 100 PIV note transferred to a new 100 PIV note leaves two commitments. Membership alone must not produce 200 PIV of voting entitlement. Existing payment validation correctly checks spent nullifiers separately from proof/anchor validity. [C12] [C13] [C16]

- Bind eligibility to one explicit block hash and height. Use the same effective snapshot across transparent and shielded funds.
- Prove spending authority, eligible value, note membership and unspentness at that snapshot. A watch-only viewing key must not grant voting authority.
- Use election/proposal-specific voting nullifiers with a proven relationship to the entitlement. Do not publish the ordinary payment nullifier as a reusable voting identifier.
- Prove conservation when splitting an entitlement into multiple voting credentials. Per-credential uniqueness without conservation is insufficient.
- Define whether a post-snapshot transfer changes rights. Under fixed-snapshot semantics the original entitlement persists for that election; newly created replacement notes must not gain a second entitlement.

The existing proof verifies a fixed statement

The pinned Groth16 Sapling verifier exposes the randomized key, value commitment, anchor and payment nullifier. It does not verify a historical non-spend proof or a proposal-specific ballot. New voting statements need new circuits or an independently reviewed proof composition. Reusing the cryptographic library is different from reusing its payment proof unchanged. [C17] [R1] [R2]

A possible extension derives an authenticated spent-nullifier set for each snapshot and proves non-membership while keeping the payment nullifier inside the proof. Current flat database lookups are not that protocol. A chain-derived authenticated representation need not necessarily add a new block-header field, but its derivation and binding must be consensus-defined. [C13]

Avoid additional privacy leaks

Publishing an exact voting weight can fingerprint holdings even when the owner is hidden. Fixed-denomination anonymous credentials can reduce this leak, with explicit conservation and batching rules. Fees, relayer choice, witness queries, timestamps and grouped submissions can still link participation. Current witness retention is bounded and needs an election-snapshot strategy. [C15]

07 / Proposal 1: Tango

Tango assigns one unit of voting weight per PIV earned from staking or masternode rewards in a cycle. An address signs a Yes/No vote and the network derives its weight from reward history. No separate monetary ticket is required. [P1]

Good points

- Entitlement is derived from chain evidence rather than a self-declared balance.
- It rewards participation in consensus and masternode services.
- A fixed reward window limits the number of reward events to index.
- It is easier to specify and explain than a multi-layer feedback system.

Problems for fully inclusive private voting

- Passive holders and entirely shielded holdings have no entitlement from those holdings under current staking rules. A small active staker with no reward in the window also has zero reward-derived weight.
- Signing from reward addresses makes ballots linkable to public reward history.
- Its seesaw-era totals and approximate 50/50 balance are obsolete. Current scheduled rewards are 6 PIV to masternodes and 4 to stakers, implying approximately 60/40 weight if copied directly. [C26]
- Summing coin stake outputs counts returned principal as well as reward. Reward extraction must distinguish staking principal, masternode payments and treasury outputs. [C19]
- Cold staking has owner and staking keys. Decide who receives governance authority; do not accidentally give a service operator the owner vote. [C20]
- The document does not complete deterministic cutoffs, late rewards, reorganisations, vote replacement, private ballots or treasury settlement.

Implementation sequence

Specify reward window and authority -> index actual eligible reward events -> freeze cycle entitlements -> implement signed votes and deterministic counting -> integrate wallet/offline owner authorization -> replace final budget approval -> test reorganisations and reward parsing -> activate by network upgrade.

Verdict: a useful incremental public governance feature, but not a solution to all-holder private participation. Expanding it to passive SHIELD holders changes its core entitlement model.

Estimate: 4-8 months for a bounded public feature under existing infrastructure; 6-10 months for deterministic binding settlement without the old approval gate. These are engineering ranges, excluding adoption delays.

08 / Proposal 2: Weighted Voting

The operative weights are 1.0 per PIV for masternode owners, 0.8 for stakers meeting a 1,000 PIV threshold, 0.65 for other stakers and 0.5 for holders. It uses one combined layer and proposes turnout/approval rules plus constitutional protection rules. Introductory example weights differ from the detailed implementation, so the latter was used for this review. [P2]

Good points

- It intends to give ordinary holders a voice and specifies concrete weights.
- One combined layer avoids some separate-chamber veto and normalisation problems.
- Advance notice, proposal thresholds and protection for important rules are useful design questions.
- The author acknowledges concentration of voting power rather than presenting it as one-person democracy.

Problems and ambiguities

- It deliberately grants different influence per coin. That is a policy choice rather than an implementation of equal holder rights.
- A wallet is a local container, not a consensus identity. References to holdings in the same wallet require a new provable grouping rule. Users can operate multiple wallets.
- Proving staking status, 1,000 PIV grouping and extra coins assigned to a masternode without leaking holdings is considerably harder than linear note eligibility.
- The stated boundary can both pass at at least 60% Yes and fail at at least 40% No. An exactly 60/40 result needs one unambiguous equality rule.
- A market-cap-dependent constitution needs a price source; a condition depending on a public admission needs human interpretation. These cannot be fully autonomous consensus inputs without extra trust.
- Higher thresholds resist some hostile changes but make minority veto and funding gridlock easier. A higher approval percentage does not establish broad turnout.

Implementation sequence

Define consensus-visible classes and assignments -> choose snapshot/history semantics -> encode rational weights with bounded integer arithmetic -> resolve equality and abstention rules -> implement private class proofs if required -> restrict autonomous constitutional rules to objective chain data -> integrate settlement and migration.

Verdict: implementable after substantial specification work, but a weak fit for equal holder influence and unnecessary complexity for a first private voting release.

Estimate: 8-14 months for a complete public-class implementation; 15-24+ months if private SHIELD class proofs and binding settlement are included. Objective constitutional parameter rules are feasible; arbitrary social conditions remain outside autonomous enforcement.

09 / Proposal 3: Votes-For-All

VFA gives each PIV, including fractional PIV, equal voting weight. It proposes mirroring balances into voting tickets on a separate chain, desktop integration, ranked choices, a status-quo option, and different approval thresholds for standard and sensitive decisions. Its foreword expressly defers technical methodology. [P3]

Good points

- Equal per-coin weight is simple and invariant under splitting a fixed balance across identities.
- All holders are intended to participate, not only service operators or reward winners.
- A fixed election snapshot is a useful starting point for preventing transfer-based duplicate entitlement.
- The status-quo option and staged implementation deserve consideration.

Problems for the proposed implementation

- A public balance mirror cannot enumerate private SHIELD holdings. Ticket issuance needs private membership, unspentness, conservation and authorization proofs.
- A second chain needs a defined validator selection, consensus security, data availability, fork handling and result-verification mechanism on PIVX. An administrator publishing the result would be an oracle.
- A snapshot blocks naive transfer/revote duplication but does not stop borrowed voting power, custody concentration, coercion or vote buying.
- Ranked voting needs exact elimination, equality, exhausted-ballot, status-quo and quorum semantics. Ranked systems are not automatically manipulation-proof.
- The fee and GUI discussion reflects 2017 conditions. A governance GUI now exists, and the current finalization fee differs from the old assumption. [C27] [C29]

Implementation sequence as written

Specify private snapshot and ticket proofs -> design and secure the voting chain -> define PIVX verification of voting-chain results -> build wallet synchronization and ticket handling -> implement exact ballot/tally rules -> integrate treasury validation -> test cross-chain forks, withholding and recovery.

Preferred adaptation

Retain equal entitlement per PIV and a fixed snapshot, but put accepted ballots or sufficient canonical proof-bearing state on PIVX itself. Start with Yes/No choices and fixed objective settlement rules. This removes the second validator set and bridge from the initial design, while leaving private eligibility as a major cryptographic task.

Verdict: strongest principle for the stated goal after redesign. The recommended single-chain version is not VFA exactly as written.

Estimate: 12-24+ months for a complete separate-chain version before the strongest private-ballot requirements; 12-18 months for the simpler single-chain adaptation with private eligibility and public anonymous choices, conditional on prototype success.

10 / Proposal 4: Pivmetheus

Pivmetheus combines network-effect, support and intelligence layers. It uses non-linear holdings weights, reward history, exponential moving averages, layer blocking, ranked comparisons and meta-votes. Candidate exponents include 0.6 and 1.66. The onramp includes two separate year-long observation stages. [P4] [P5]

Good points

- It explicitly considers minority protection, voter apathy, concentration and long-term incentives.
- A status-quo option and explicit treatment of negative feedback can improve the decision model.
- Reward history can provide objective evidence of network service.
- It recognizes that parameters require observation rather than assuming a final perfect setting.

Major unresolved issues

- An account is not a person. Concave per-account weights reward splitting unless the full system supplies a demonstrated countermeasure. A convex companion layer changes incentives but does not prove Sybil resistance.
- The no-dancing moving average delays new balances and account splits; aged splits still mature. Private transfers and account linking make the historical model harder to implement.
- Small-holder and large-holder group membership cannot be reliably inferred from private addresses alone. Account distribution is not a measurement of human wealth distribution.
- Layer vetoes and override rules need exact definitions and adversarial analysis. They create potential gridlock surfaces alongside intended minority protection.
- Fractional powers and continuous-time formulas must become deterministic fixed-point consensus arithmetic, with exact rounding, time source, caps and update rules.
- Tuning thresholds or interpreting market signals must not depend on a privileged analyst. Optional identity and ZK work is mentioned, not specified as an implemented solution. [P6]
- Pruning ballot evidence after a year weakens later independent replay unless sufficiently strong proofs and availability guarantees replace it.

Implementation sequence

Write one executable mathematical specification -> test adversarial account partitions and layer dynamics -> prove or revise the claimed incentive properties -> choose private identity/grouping assumptions -> build deterministic history and arithmetic -> integrate proofs, ranking, settlement and no-admin parameter changes -> conduct staged observation.

Verdict: research programme rather than the best first implementation for this goal. Its fairness claims need evidence beyond the proposal narratives.

Estimate: 18-30+ months for a credible initial implementation, with low confidence before model validation. The proposed observation programme adds about two years of staged learning; it is not included as completed merely because software launches.

11 / Comparison against the goal

Criterion	Tango	Weighted	VFA	Pivmetheus
Every holder intended	No	Yes	Yes	Broad, with thresholds
Equal weight per PIV	No: reward based	No: classes	Yes	No: non-linear layers
Address-split neutrality	Reward events limit it	Class/grouping dependent	Yes, if entitlement conserved	Unproven for full model
SHIELD entitlement fully specified	No	No	No	No
Private ballot protocol supplied	No	No	No	No
Extra voting chain	No	Not required	Proposed	Optional
Tuning/research burden	Lower	Medium/high	Medium; higher cross-chain	Very high
Best use	Incremental service voting	Explicit class privileges	Foundation after redesign	Long-term research

No proposal automatically guarantees permissionless proposal admission, complete canonical tallying, privately eligible SHIELD balances, no privileged override and no separate ratification. Those properties depend on the actual protocol and Core integration.

Reproducible component checks

Check	Result	Interpretation and limitation
Linear holdings split into 100 accounts	1.00x total weight	Splitting alone does not improve conserved linear entitlement.
100-way split under a 0.6 power	6.31x concave-component weight	Only that component; not a simulation or exploit proof of the complete Pivmetheus system.
EMA at 4, 8, 12 months	63%, 86.31%, 94.93%	The proposed time constant delays but does not identify account owners.
Tango with 0.001% active stake share	About 64.9% chance of zero rewards in 43,200 trials	Illustrative independent proportional-reward model, not measured PIVX outcomes.
60% approval at 10% turnout	Yes can be 6% of eligible weight	Approval and turnout are different quantities; exact boundary must be specified.

The supplied `model_checks.py` reproduces these calculations. It deliberately does not claim to implement or validate any full CDG protocol.

12 / Recommended architecture and privacy choices

Use PIVX consensus as the single canonical bulletin board and execution layer. Keep eligibility, ballot acceptance, tally completeness and treasury settlement independently verifiable.

Core state machine

Proposal admitted -> eligibility snapshot fixed -> credentials registered -> ballots accepted before a fixed cutoff -> canonical tally or tally proof finalized -> execution delay -> bounded treasury payments. State transitions, nullifiers and undo information must be replayable across restart, reindex and reorganisations.

- Any qualifying holder can register and submit valid ballots; any relayer can carry them.
- A proof authenticates eligible spending authority and conserved weight without exposing payment nullifiers or requiring a viewing-key upload.
- No masternode operator vote ratifies the result. Nodes verify the result mechanically.
- No spork or admin key can replace a tally, exclude a valid class or disable the new payout rules.
- If an aggregator is used, anyone can produce the proof. The proof must cover the canonical accepted ballot set, not a subset chosen by the aggregator.

Choose the ballot privacy model explicitly

Model	Trust/liveness implications	Fit
Anonymous public choices, common weight denominations	No decryption authority. Private eligibility remains difficult; timing and grouping still leak.	Best bounded first target if public anonymous choices are acceptable.
Commit/reveal	Voters must return to reveal; selective non-reveal can bias results. Revealed weights/choices are public.	Not permanently secret voting. Needs explicit non-reveal rules.
Threshold-encrypted tally	Verifiable counting, but sufficient keyholders can collude on privacy or withhold decryption.	Decentralised only under an explicit threshold assumption; fails a strict no-special-committee requirement.
Committee-free secret/self-tallying design	Requires a reviewed protocol for dropout recovery, scalability and fairness; time-lock approaches add timing/computational assumptions.	Research-gated option for the strictest requirement.

Current Zcash voting work is useful reference material, but it uses Orchard-era components, and the inspected vote-chain documentation has coordinator approvals and threshold decryption. It is not a drop-in Sapling solution or evidence that this project has no privileged actors. These external systems were not audited here. [\[E1\]](#) [\[E2\]](#) [\[E3\]](#) [\[E4\]](#)

Self-tallying voting with zk-SNARKs is also a research direction. Its existence does not establish a reviewed, scalable, dropout-tolerant PIVX implementation. [\[E5\]](#)

13 / Minimum protocol contract before coding

The following is a specification checklist, not a completed cryptographic construction. An implementer must turn these statements into a reviewed circuit/composition and deterministic consensus rules.

Object / transition	Minimum binding and invariant
Proposal	Immutable proposal ID; payout script and schedule; requested value; admission fee; cycle; text/document commitment. Changing a payee or value requires a new authorization. A content hash does not itself guarantee that voters can retrieve the document.
Snapshot	Explicit chain identifier, protocol version, block hash and height. Transparent unspent outputs, Sapling note commitments and spent-nullifier representation must describe the same chain state. Define treatment of collateral, locked and immature funds.
Registration proof	Privately establish eligible ownership/spending authority and value at the snapshot; establish note membership and non-spend; bind all public roots and scope; reject duplicate input entitlement; conserve value into voting credentials.
Registration nullifier	Uniquely consume the same eligible input for credential issuance within the election, while leaving its payment nullifier hidden. The derivation and uniqueness argument require cryptographic review.
Credential	Commit to a voting authorization key, denomination/weight, election and randomness. Credential totals cannot exceed the proved eligible value. Split and merge rules must preserve this invariant.
Ballot proof	Prove authorized use of an accepted credential; bind proposal, choice, election, chain and protocol version; enforce that choices are in range; reveal a proposal-scoped voting nullifier. Define replacement versus first-vote-wins exactly.
Canonical accepted set	Every valid included ballot before cutoff enters the tally under the replacement rule. An aggregation proof must bind the full accepted set/root; anyone must be able to aggregate or verify directly.
Settlement	Use exact integer arithmetic, equality rules, treasury cap and deterministic funding order. Payouts must match approved amounts/scripts and occur once under the approved schedule. Reorg undo must reverse every affected state transition.

Do not assume all transparent scripts have identical authorization. Cold staking, multisignature, hardware/offline keys and future output types need an explicit eligibility/signing rule. Count masternode collateral only once as coin value; do not silently add a service-class bonus in the equal-PIV design. [\[C20\]](#)

For privacy, measure the entire registration-to-fee-to-ballot path. A proof hiding a note does not prevent correlation if a wallet submits every credential together or pays from a recognizable public address. Small denominations improve mixing opportunities but increase proof count and chain load.

14 / Step-by-step delivery: define and prove

Step	Concrete work	Exit criterion / owner
1. Adopt requirements	Decide coin vs human weighting; ownership, choice and amount privacy; custody; cold-staking authority; permitted residual trust.	A community-reviewable requirements document. Product/protocol owner.
2. Freeze consensus rules	Specify snapshot hash/height, proposal admission, fees, deadlines, abstentions, equality, revotes, quorum denominator, ties and execution limits.	Independent implementations produce identical outcomes on test vectors. Core and protocol engineers.
3. Specify cryptography	Define public inputs, private witnesses, spent-set proof, authorization, domain separation, registration conservation and voting nullifiers.	Independent cryptographer reviews a complete statement and security argument.
4. Build a small prototype	Generate and verify real Sapling-derived entitlements. Test spent notes, duplicate registration, split credentials, malformed proofs and wrong snapshots.	All adversarial cases reject; client time, verifier time, memory and proof bytes measured. ZK engineer.
5. Decide private tally architecture	Choose public choices or demonstrate a secret tally protocol with a clear dropout/withholding recovery rule.	No hidden coordinator key or unreviewed decryption assumption. Protocol and cryptography reviewers.
6. Model participation and attack costs	Backtest plausible turnout, custody concentration, lending, split/merge behaviour and treasury exposure. Resolve quorum policy.	Documented scenarios and sensitivity analysis, not a claim that a higher threshold makes capture impossible.

Early funding milestone

Fund an 8-12 week prototype and specification milestone before committing to a production schedule. The most important demonstration is that an eligible SHIELD holder can obtain correct weight, while an ineligible/spent entitlement and every duplicate are rejected, without exposing their ordinary payment nullifier or exact holding.

The prototype must also show how a second implementation reconstructs the snapshot roots. A server returning an unverified root is not a decentralised eligibility solution. Client queries must not reveal the hidden nullifier to the server; full local reconstruction remains a fallback, with optional privacy-preserving retrieval. [\[E4\]](#)

If committee-free secret voting is mandatory and that milestone cannot supply a reviewed tally protocol, stop at a research result rather than relabelling a committee-based service as the completed goal.

15 / Step-by-step delivery: implement and activate

Step	Concrete work	Exit criterion / owner
7. Implement chain state	Version governance objects; retain snapshot roots and canonical ballots; persist nullifiers; implement ConnectBlock/DisconnectBlock undo and reindex.	Independent nodes agree after cache deletion, restart and supported reorganisations. Core engineers.
8. Integrate proofs and resource limits	Add Rust/C++ verification interfaces, validated keys/parameters, cheap rejection checks, proof caches, fee and per-block verification budgets.	Malformed or zero-weight spam cannot impose unbounded verification work. Core/ZK engineers.
9. Build wallet and relay support	Local witnesses/proofs, offline or hardware signing plan, cold-staking owner rules, private fee path and multiple independent relayers.	A holder can vote without giving a service spending/viewing keys; a failed relayer can be bypassed. Wallet engineer.
10. Replace treasury authorization	Use the canonical tally, deterministic funding order, explicit ties, budget caps, payment schedule and execution delay. Remove operator discretion.	Wrong tally, omitted ballots, over-budget and duplicate payments reject. Core/QA.
11. Sunset privileged paths	Height-gate old cache/masternode/spork behaviour away from the new rules. Preserve historical validity and a deterministic migration.	A valid legacy control message cannot alter a new election or payout. Core and independent review.
12. Test, audit and deploy	Run adversarial tests, multi-node testnet, reproducible builds, wallet recovery tests and independent cryptographic/consensus review. Coordinate release and activation.	Published acceptance results and adoption readiness before activation. Release/QA/operators.

Do not confuse deployment with agreement

Agree the transition through the existing community process and a concrete protocol proposal. A treasury funding vote can pay for development; it does not itself activate a protocol. Upgraded validators and supporting services must adopt compatible rules. Existing upgrade machinery is height-based. [\[C24\]](#)

Use a forward activation height after adequate distribution and testnet operation. An execution delay provides time to observe a result but must not give a named operator discretionary veto. An emergency mechanism, if retained, needs bounded rules and explicit trust implications.

First scope should be treasury and objective allowlisted parameters. Treat manifesto changes, staffing decisions and arbitrary software replacements as social decisions unless their execution is explicitly encoded and adopted by nodes.

16 / Core work packages and interfaces

Work package	Existing files / reuse	New behaviour required
Eligibility and history	coins.cpp; sapling_txdb.cpp; validation.cpp [C12] [C13] [C16]	Deterministic snapshot representations across pools; private non-spend proof witnesses; reorg-safe history.
Cryptography	Cargo.toml; rustzcash.rs; pinned Sapling circuits [C17] [R1] [R2]	New reviewed voting circuit or composition; authorization, conservation, scope binding, setup/keys and FFI.
Ballot admission	budgetvote.h/cpp; budgetmanager.cpp [C1] [C3]	New proof-bearing object; election and proposal identifiers; duplicates, revotes, fees and deadlines.
Governance state	budgetdb.cpp; validation connect/undo [C9] [C16]	Canonical state derived from chain records, not a network-only budget cache.
Tally and execution	budgetproposal.cpp; finalizedbudget.cpp; masternode-payments.cpp [C4] [C5] [C7] [C8]	Complete deterministic results; no second political approval; bounded payout validation and undo.
Authority removal	spork.cpp; chainparams.cpp; upgrades.cpp [C21] [C22] [C23] [C24]	Explicit migration and removal of privileged override for the new voting system.
Wallet and RPC	saplingscriptpubkeyman; rpc/budget.cpp; governance model [C14] [C15] [C27]	Local private proofs, stable snapshot witnesses, intent display and private fee/relay handling.
Consensus limits	consensus.h and policy [C25]	Bound voting proof CPU, memory, payload size, registration outputs and accepted ballots.

Reuse with care

Do not change the existing payment proof statement merely to add governance. A separate voting extension reduces the chance of unintentionally changing monetary privacy or conservation. Reuse primitives and wallet data structures after review, while keeping separate domain identifiers, parameter versions and nullifier namespaces.

If a new Groth16 circuit is chosen, its proving/verifying parameters and setup must be addressed. A transparent proof system can avoid a circuit-specific secret setup, but brings different performance and integration tradeoffs. The choice requires measurements rather than assuming that all zk-SNARKs are interchangeable.

17 / Required adversarial acceptance tests

Property	Tests before mainnet
Entitlement conservation	Already-spent note at snapshot; same note registered twice; many credentials exceeding input weight; transparent-to-shielded transfers across the snapshot; immature and zero-value inputs.
Authorization and privacy	Viewing-key-only claimant; wrong spending key; substituted recipient/key; exposed payment-nullifier correlation; public amount fingerprint; fee and witness-query linkage.
Scope and replay	Replay on a different chain, cycle, proposal, snapshot or protocol version; re-randomized duplicate proof; repeated credential with different choices.
Canonical outcomes	Omitted accepted ballots; injected ballots; two conflicting tally proofs; exact threshold equality; ties; exhausted ranked ballots; abstentions; incorrect funding order.
Liveness and censorship	Offline coordinator or aggregator; relay outage; unavailable witnesses; decryption-share withholding; selective non-reveal; sustained producer censorship under stated consensus assumptions.
State integrity	Restart; deleted governance cache; reindex; supported-depth reorg across snapshot, close and payout; pruned node bootstrap; repeated/disconnected payment.
Resource and fees	Large proofs, many input notes, zero-weight spam, cheap invalid-object floods, expensive verification floods, fee sponsorship abuse and maximal blocks.
Treasury safety	Budget overrun; arithmetic overflow; duplicate funding; payee mutation after approval; execution before delay; unexpected continuation of multi-cycle grants.
No privileged bypass	Old spork messages, alternate budget modes, operator finalization and stale tier-two state cannot change the new accepted outcome.
Client independence	Two independent verifiers on separate platforms obtain the same result; offline signer shows the intended proposal and scope.

Tests executed during this audit validate existing PIVX behaviour. They do not substitute for this new-protocol matrix. Passing existing nullifier tests does not prove a new voting-nullifier construction is secure.

For secret ballots, require an explicit claim about choice privacy before close, privacy after tally, receipt-freeness and coercion. Encryption alone does not prevent a voter proving or selling their vote; no proposal here completes coercion resistance.

18 / Effort, staffing and uncertainty

Scope	Elapsed planning range	Assessment
Tango public reward voting	4-8 months	Bounded feature; does not meet passive SHIELD-holder inclusion.
Tango with deterministic binding settlement	6-10 months	Requires authority/state migration; still reward-based.
Weighted public classes	8-14 months	Class definitions and constitutional semantics need resolution.
Weighted with private classes	15-24+ months	Additional private grouping/history proofs; low confidence until specified.
VFA as a separate voting chain	12-24+ months	Additional validator security, bridge/result verification and private ticket protocol.
Pivmetheus initial system	18-30+ months	Research-heavy; approximately two years of staged observation remains additional.
Recommended single-chain linear design	12-18 months	Private eligibility and publicly countable anonymous choices; prototype-gated.
Strict committee-free permanent ballot secrecy	18-30+ months, provisional	Research gate; a fixed delivery promise is not defensible before demonstrating the tally protocol.

Team and work distribution

A credible production effort needs two Core/consensus engineers, a ZK/protocol engineer, a wallet/client engineer, QA/release support and independent cryptographic/consensus reviewers. Roles can overlap, but independent review should not be performed solely by the implementer. A specialist shortage can dominate elapsed time.

For the recommended bounded design, allow approximately 45-80 person-months across engineering, testing and review, as a coarse capacity range rather than a quote. Some work runs concurrently. A strict secret-tally research track may materially exceed this range. Funding and network adoption delays are excluded.

Milestones rather than one large promise

Weeks 1-4: requirements, threat model, exact rules and public review. Months 2-3: eligibility prototype and adversarial vectors; stop or revise if privacy and conservation cannot be demonstrated.

Months 3-8: chain state, proofs and wallet integration. Months 7-12: settlement, control-path migration and performance testing. Months 10-18: independent review, remediation, public testnet, reproducible releases and activation readiness.

19 / Activation, governance and release gates

Consensus required

For an advisory external poll, PIVX consensus need not change. For independent holder voting that mandates treasury payouts, the project should plan a coordinated consensus upgrade, likely a hard fork under this design. Existing nodes need not agree with new eligible voters or payment authorization. A finalized result on another chain is not automatically valid to PIVX. [C7] [C8] [C24]

There is no generic 51%, 60% or 75% upgrade trigger in the inspected code. Approval thresholds in the CDG papers are proposed governance policies. Activation and adoption require agreement on an actual implementation, its consensus rules and upgrade height.

Release gates

- Published protocol and threat model define all privileged roles and residual assumptions.
- Two verifiers agree on cross-platform vectors and complete canonical tallying.
- Private entitlement prototype demonstrates unspentness, conservation and non-linkability goals with measured performance.
- Wallet recovery, cold staking, custody boundaries and fee privacy are documented and tested.
- Supported reorganisations, reindex and history migration give identical results.
- New-governance outcomes cannot be bypassed by old masternode approval or spork paths.
- Independent cryptographic and consensus reviews are complete; findings are remediated and regression-tested.
- Testnet completes multiple full election/funding cycles with adverse networking and operator outages.
- Release adoption and service readiness support the planned activation; users retain the ability to assess the rules.

Final recommendation

Adopt equal coin entitlement and one canonical chain. Prove private eligibility. Make tallying and settlement permissionless and deterministic. Choose a clearly specified ballot-secrecy model before estimating the final release.

Tango is a reasonable incremental improvement if service participation is the goal. It is not the shortest route to universal private-holder voting. Weighted Voting and Pivmetheus introduce policy and verification complexity that should be justified separately, rather than included in the first release by default.

The first funded deliverable should be a reviewable specification, a real eligibility proof prototype, adversarial vectors and performance measurements. That gives the community a concrete basis for approving or rejecting the longer implementation effort.

20 / Core source references

Inline reference IDs are clickable. Core and CDG source links are pinned to the audited commits; external research links are reference material, not audited dependencies.

[C1] Plaintext governance ballot serialization

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/budget/budgetvote.h#L18

[C2] Public vote JSON fields github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/budget/budgetvote.cpp#L53

[C3] Proposal-vote eligibility and signatures

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/budget/budgetmanager.cpp#L1089

[C4] Proposal passing threshold

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/budget/budgetproposal.cpp#L200

[C5] Final budget voting and operator approval

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/budget/budgetmanager.cpp#L561

[C6] Final budget payment thresholds

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/budget/budgetmanager.cpp#L693

[C7] Block payee validation and spork gates

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/masternode-payments.cpp#L227

[C8] Expected payee and amount matching

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/budget/finalizedbudget.cpp#L350

[C9] Separately persisted budget cache

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/budget/budgetdb.cpp#L17

[C10] Sapling transaction and staking restrictions

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/sapling/sapling_validation.cpp#L19

[C11] Sapling proof checks

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/sapling/sapling_validation.cpp#L180

[C12] Shielded anchor and spent-nullifier checks

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/coins.cpp#L550

[C13] Nullifier database representation

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/sapling/sapling_txdb.cpp#L7

[C14] Wallet-local note decryption

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/sapling/saplinscriptpubkeyman.cpp#L514

[C15] Bounded wallet witness cache

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/sapling/saplinscriptpubkeyman.h#L18

21 / Core and CDG references

Inline reference IDs are clickable. Core and CDG source links are pinned to the audited commits; external research links are reference material, not audited dependencies.

[C16] Block-committed Sapling root

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/validation.cpp#L1604

[C17] Pinned Rust proof dependency github.com/PIVX-Project/PIVX/blob/bd8a672d1d/Cargo.toml#L22

[C18] Payment-nullifier derivation github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/sapling/note.cpp#L46

[C19] Coinstake output creation and principal

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/wallet/wallet.cpp#L3326

[C20] Cold-staking owner and staker extraction

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/script/standard.cpp#L56

[C21] Network control switches github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/spork.cpp#L15

[C22] Spork signature authority github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/spork.cpp#L148

[C23] Mainnet activation and spork keys

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/chainparams.cpp#L252

[C24] Height-based upgrade activation

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/consensus/upgrades.cpp#L82

[C25] Block size, shielded size and reorg constants

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/consensus/consensus.h#L13

[C26] Reward schedule and masternode allocation

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/validation.cpp#L816

[C27] Wallet governance interface

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/qt/governancemodel.cpp

[C28] Proposal ordering and fee-hash tie break

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/budget/budgetproposal.h#L149

[C29] Proposal and finalization fees

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/budget/budgetproposal.h#L15

[C30] Spork-gated quorum ChainLocks verification and enforcement

github.com/PIVX-Project/PIVX/blob/bd8a672d1d/src/llmq/quorums_chainlocks.cpp#L86

[R1] Pinned Sapling circuit public inputs

github.com/PIVX-Project/librustzcash/blob/e7662b23d1/zcash_proofs/src/sapling/verifier.rs#L73

22 / Proposals, runtime and research references

Inline reference IDs are clickable. Core and CDG source links are pinned to the audited commits; external research links are reference material, not audited dependencies.

[R2] Pinned Sapling Spend circuit

github.com/PIVX-Project/librustzcash/blob/e7662b23d1/zcash_proofs/src/circuit/sapling.rs#L51

[P1] Tango proposal github.com/PIVX-Project/CDG/blob/4806f817d8/Tango.md

[P2] Weighted Voting System proposal [github.com/PIVX-Project/CDG/blob/4806f817d8/Weighted Voting System.md](https://github.com/PIVX-Project/CDG/blob/4806f817d8/Weighted_Voting_System.md)

[P3] Votes-For-All proposal [github.com/PIVX-Project/CDG/blob/4806f817d8/Votes-For-All Consensus Proposal.md](https://github.com/PIVX-Project/CDG/blob/4806f817d8/Votes-For-All_Consensus_Proposal.md)

[P4] Pivmetheus strategic weight analysis

[github.com/PIVX-Project/CDG/blob/4806f817d8/Pivmetheus/Strategic Weight Analysis.md](https://github.com/PIVX-Project/CDG/blob/4806f817d8/Pivmetheus/Strategic_Weight_Analysis.md)

[P5] Pivmetheus implementation and staged observation

github.com/PIVX-Project/CDG/blob/4806f817d8/Pivmetheus/Implementation.md

[P6] Pivmetheus optional ZK and identity work

[github.com/PIVX-Project/CDG/blob/4806f817d8/Pivmetheus/Discussion and Reference.md](https://github.com/PIVX-Project/CDG/blob/4806f817d8/Pivmetheus/Discussion_and_Reference.md)

[T1] Executed official release github.com/PIVX-Project/PIVX/releases/tag/v5.6.1

[T2] Release-matched functional tests github.com/PIVX-Project/PIVX/tree/af60f19642/test/functional

[E1] Zcash Orchard voting integration issue, March 2026

github.com/zcash/zcash-swift-wallet-sdk/issues/1661

[E2] Zcash voting cryptographic library; reference only github.com/valargroup/zcash_voting

[E3] Vote chain coordinator and threshold-decryption design; reference only

github.com/valargroup/vote-sdk

[E4] Private nullifier non-membership retrieval; reference only github.com/valargroup/vote-nullifier-pir

[E5] ElSheikh and Youssef, Open Vote Network with zk-SNARKs, 2022 arxiv.org/abs/2203.03363