Privacy Bank Chain

Private money that works for you.

Whitepaper technique — v1.8 — août 2026

Privacy Bank Chain (PBC)

Whitepaper v1.8

Complete protocol specification — monetary model, term deposits, claims, early withdrawal, on-chain bond market, trustless inheritance, hybrid post-quantum cryptography

2026-08-09

Abstract

Privacy Bank Chain (PBC) is a privacy-focused Proof-of-Work blockchain that embeds native banking primitives directly into consensus: tiered time-locked deposits, claim-based reward distribution through deterministic global indexes, network-fee redistribution, an insurance fund with automatic deflationary burn, early term withdrawal with a fixed penalty, an atomic on-chain secondary market that makes every deposit a freely tradable bearer bond, and a trustless inheritance mechanism ("dead man's switch") protected by a consensus gate.

All financial identities are protected by CryptoNote primitives (ring signatures, stealth addresses, RingCT). Operation amounts required by consensus accounting are public — a hybrid-transparency model: observers see *how much*, never *who*. On top of the classical stack, PBC layers NIST-standardised post-quantum primitives (ML-DSA-65, ML-KEM-768) in hybrid mode on every public spend-authority path.

There are no smart contracts, no oracles, no admin keys and no third-party custody: every rule in this document is compiled, deterministic C++ validated identically by every node.

> **Block-time convention.** All consensus durations are defined in **days** and converted to blocks as `blocks = days × 86400 / DIFFICULTY_TARGET_V2`. The implementation targets **60-second blocks** (1 440 blocks/day); every block-denominated figure below derives from that constant and rescales automatically if the target ever changes.

---

1. Design Principles

1. Deterministic consensus rules
2. Integer-only arithmetic (no floating point; 128-bit intermediates)
3. RingCT equation preservation
4. Reorg-safe state transitions (strict apply/pop symmetry, explicit undo records)
5. Pool accounting symmetry and provable solvency
6. No hidden minting or leakage; full supply conservation
7. Hybrid transparency: amounts of banking operations are public, identities never are
8. No smart contracts, no oracles, no admin keys: every rule is compiled consensus code

---

2. Base Protocol

Parameter	Value
Codebase	CryptoNote / Monero-derived, C++
Privacy	RingCT (CLSAG), ring signatures, stealth addresses; Dandelion++ transaction relay at the network layer
Proof of Work	**RandomX v2** (`rx/pbc` variant), CPU-only, ASIC-resistant, Stratum-pool compatible
Block target	60 seconds (`DIFFICULTY_TARGET_V2 = 60`) → 1 440 blocks/day
Difficulty	Two-phase bootstrap at chain initialisation, then per-block LWMA retargeting
Atomic units	1 PBC = 10^{12} atomic units (12 display decimals)
Address prefixes (mainnet)	`Pbc…` standard, `Pbc…` v2 (PQC-committed), `PmK…` integrated, `Pw2…` subaddress
Network name	`pbcchain`
Hard forks	v17 genesis rules → v21 (current consensus, incl. inheritance sweep gate) → v23 (mandatory PQC spend authority)

Miners interact with a standard coinbase structure; all banking features are wallet-level transaction types validated by consensus (Appendix A). Any Stratum pool works unmodified. User-side tooling ships with the node: CLI wallet, wallet-RPC service, and an optional local web interface that connects exclusively to the user's own node — keys never leave the machine.

---

3. Monetary Model

3.1 Supply and Emission

```
MONEY_SUPPLY = 18 446 744 PBC            (18 446 744 × 10¹² atomic units)
base_reward  = (MONEY_SUPPLY − already_generated) >> ESF
ESF          = 22 − (target_minutes − 1)  = 22 at 60-second blocks
tail emission = 0                          (FINAL_SUBSIDY_PER_MINUTE = 0)
```

- Genesis-era reward at the current target: **≈ 4.398 PBC/block ≈ 6 333 PBC/day**, decaying smoothly as supply is emitted.

- **The supply is strictly finite.** Unlike Monero and other CryptoNote chains, PBC has no perpetual tail emission; once `already_generated ≥ MONEY_SUPPLY`, block subsidy is zero and miner revenue comes from transaction fees. Combined with the insurance-cap burn (§4.4), PBC is deflationary at the margin.
- Total coins introduced per block = `R + fee_summary`, where `fee_summary = Σ(tx.fee)`.

3.2 Per-Block Split (exact integer arithmetic)

Every block reward `R` and fee total `F` are split deterministically (per-mille, remainder-absorbing so conservation is exact):

Recipient	Share of R	Share of F	Form
Miner	91.0 %	50 %	Coinbase outputs (vested, §3.3)
Development Fund	2.0 %	—	Coinbase output to the dev-fund address
Fee Pool	remainder of 7 % pool share (≈ 3.5 %)	50 %	Virtual pool balance
Deposit Pool	25⁄70 of pool share (= 2.5 %)	—	Virtual pool balance
Insurance Pool	10⁄70 of pool share (= 1.0 %)	—	Virtual pool balance

Computation order (`pbc_compute_block_split`): `miner_R = ⌊R·910/1000⌋`, `dev_R = ⌊R·20/1000⌋`, `pools_R = R − miner_R − dev_R` (absorbs rounding dust); the pool sub-split gives Deposit `⌊pools_R·25/70⌋`, Insurance `⌊pools_R·10/70⌋`, Fee Pool the remainder. Fees: `miner_F = ⌊F·500/1000⌋`, Fee Pool the remainder. All intermediates use unsigned 128-bit arithmetic; conservation is asserted at every step:

```
miner_R + dev_R + deposit_share + insurance_share + fee_share = R
miner_F + pools_F = F
```

Development-fund transparency by construction. The dev-fund allocation is verified by every node: the fund's *view* key is published in the source code, which is both a validation requirement (each node derives the fund output to check that the coinbase pays exactly 2 %) and a public-accountability feature — anyone can audit every coin the fund receives, at any time. The fund's spend key is never part of the code.

3.3 Miner Reward Vesting (anti-dump)

From hard fork v19, every miner coinbase reward is split into **4 equal outputs** with staggered unlock heights:

Output	Unlock delay
1	1 440 blocks (~1 day)
2	43 200 blocks (~30 days)
3	86 400 blocks (~60 days)
4	129 600 blocks (~90 days)

Vesting is enforced by consensus on the coinbase itself; it cannot be bypassed by pool software. Sell pressure from freshly minted coins is structurally smoothed.

---

4. Pool Architecture, Reward Indexes and Fee Policy

PBC maintains deterministic, reorg-reversible virtual pools: **Deposit Pool**, **Fee Pool**, **Insurance Pool**, plus global accounting state (`total_locked_in_deposits` , `deposit_sum_weights` , `cumulative_fees` , `total_destroyed` , `total_vested_outputs`).

4.1 Global Reward Indexes

Depositor rewards are not paid per block; they accrue through two monotonic global indexes (deposit index and fee index), updated at each **distribution period** boundary:

```
 PBC_DISTRIBUTION_PERIOD = 1 440 blocks (~1 day)
index_scale             = 10¹⁸ (PBC_SCALE)
period inflow           = pool balance accumulated over the period (after LSM, §4.2)
index += inflow × PBC_SCALE / Σ_weights
reward(deposit) = weight × (index_now − index_at_entry) / PBC_SCALE
```

Solvency is guaranteed by construction: the sum of all claimable rewards over a period equals exactly the pool delta distributed for that period (`Σ rewards = ΔP`). Indexes are

monotonic (enforced by consensus assertions) — accrued gains never decrease. No fixed rate is ever promised; a fixed rate would be an implicit debt the chain cannot guarantee.

4.2 Locked Supply Multiplier (LSM)

To keep the system stable when a very large share of the circulating supply is locked, index inflows are dampened above a threshold:

```
locked_ratio = total_locked / circulating_supply   (per-mille, clamped to [0, 1000])
if locked_ratio ≤ 600 ‰ (60 %):    full inflow distributed
else:                               inflow rate reduced linearly,
                                    floored at 100 ‰ (10 %)
```

Undistributed inflow remains in the pool for later periods. LSM prevents a reflexive "lock everything" spiral from starving on-chain liquidity while preserving long-term claimability.

4.3 Dynamic Fee Floor

The network's minimum fee rises with adoption of the savings layer:

```
min_fee = base_fee_per_byte × tx_size × (1000 + locked_ratio) / 1000
          capped at 2× (fee_multiplier ≤ 2000 ‰)
```

As more supply is locked in deposits, the fee floor climbs (up to double), and since 50 % of every fee flows to the Fee Pool, depositors' fee-based yield mechanically strengthens with adoption. Computed in 128-bit integer arithmetic like every other consensus quantity.

4.4 Insurance Pool: Subsidy and Deflationary Burn

- **Funding**: 1 % of every block reward, plus 100 % of early-withdrawal penalties (§7).
- **Subsidy (safety net)**: at period boundaries, if the Deposit Pool falls below its floor (`PBC_MIN_DEPOSIT_POOL = 100 PBC`), the Insurance Pool tops it up, capped per period at `insurance_balance × 100/1000` (10 %). Automatic, deterministic, no governance.
- **Cap and burn (deflation)**: the Insurance Pool is capped at **184 467 PBC — exactly 1 % of the maximum supply**. Any excess above the cap is added to `total_destroyed` and permanently removed from circulating supply, every block, by consensus. No manual or discretionary burn exists anywhere in the protocol.

4.5 Circulating Supply Accounting

```
circulating = already_generated + cumulative_fees − total_destroyed
```

Consensus asserts at every block: all pools non-negative, pools ≤ circulating, `total_locked_in_deposits ≥ 0`, `deposit_sum_weights ≥ 0`, destroyed ≤ supply base. Cross-node state hashes must match.

---

5. Term Deposits

5.1 Parameters

Tier	Duration	Lock (blocks)	Weight multiplier
1	30 days	43 200	1.0×
2	90 days	129 600	1.3×
3	180 days	259 200	1.6×
4	270 days	388 800	1.8×
5	365 days	525 600	2.0×

- Minimum deposit: **10 PBC**. Maximum **10 active deposits per address**. No creation fee beyond the standard network fee.
- **Weight is linear**: `weight = amount × multiplier`. Splitting a large deposit into many small ones can never increase total weight (anti-split by construction).
- The **principal is locked at the UTXO level** until maturity (`unlock_height`); rewards remain claimable at any time. Consensus bounds `unlock_height ≤ inclusion_height + tier_blocks + 256`, preventing a modified wallet from inflating its unlock height to farm rewards beyond its tier.
- Each deposit carries the owner's spend public key and an Ed25519 ownership signature (`PBC_DEPOSIT_OWNER_V1` domain separation); deposit records are stored in LMDB, keyed by `deposit_id`, with global totals tracked.
- Reward eligibility requires `unlock_height > current_height` (an expired deposit stops earning; the principal simply becomes spendable).

5.2 Claims

A CLAIM transaction extracts accrued rewards (`weight × Δindex / SCALE`, deposit index + fee index together) without touching the deposit. Claims update the deposit's entry indexes, are fully reversible under `pop_block`, and — as a deliberate security property — **do not count as proof-of-life for inheritance** (§9.4), because claims are permissionless.

---

6. Liquidity Before Maturity — Two Exits

A depositor who needs liquidity before maturity has two consensus-native options:

1. **Early Term Withdrawal** (§7): immediate exit, fixed 2 % penalty to the Insurance Pool — *protocol-ready; CLI-only for now, not yet exposed in the wallet UI*.
2. **Bond-market sale** (§8): sell the deposit at a freely chosen price; no penalty, no confiscation.

---

7. Early Term Withdrawal (TX_TERM_WITHDRAW)

> **Availability (v8.2.11):** the mechanism is **protocol-ready** — implemented and enforced by consensus on every node, and available in the CLI wallet. It is **not yet exposed in the GUI wallet / web interface**; until it is, users who need liquidity before maturity exit via the bond market (§8). Exposure in the wallet UI is planned for a future release.

7.1 Penalty Formula (consensus rule)

```
penalty  = ⌊amount × 20 / 1000⌋      (2 %, uint64 atomic units, no floating point)
returned = amount − penalty
```

The returned output must be public (mask = 0), located at output index 0, with commitment `returned × H`, and pass BP+ verification.

7.2 Fee Semantics

```
fee_total = penalty + network_fee
real_fee  = fee_total − penalty
```

Distribution: Miner 50 % of `real_fee`; Fee Pool 50 % of `real_fee`; Insurance Pool 100 % of `penalty`. `cumulative_fees` increases by `fee_total`, preserving the RingCT equation.

7.3 Eligibility

- Allowed only while `current_height < unlock_height` (after maturity the UTXO unlocks by itself; a withdraw TX is invalid).
- CLAIM + WITHDRAW on the same deposit in the same block is invalid; double-withdraw in a block is invalid; a withdraw conflicts in-mempool with a pending market transfer of the same deposit.
- From hard fork v23, a valid **ML-DSA-65 (Dilithium) co-signature** by the owner's registered PQC key is mandatory (§10.2).

7.4 Conservation and Reorg Safety

For a block containing a withdraw: new coins = `base_reward + fee_summary`; distribution = `Miner (base_reward + 50 % real_fee) + FeePool (50 % real_fee) + Insurance (penalty)` — sums exactly, no inflation or leakage. Undo data (penalty amount, packed deposit record) allows `pop_block` to restore the deposit, reverse pool deltas, delete undo keys and recompute the split symmetrically. Insurance overflow triggered by the penalty is re-applied post-withdraw with the delta merged for reorg symmetry.

---

8. Secondary Deposit Market — an On-Chain Bond Market

A PBC term deposit is, economically, a **private bearer bond**: a principal, a maturity, a yield. The secondary market makes it tradable — fully on-chain, atomic, with no intermediary, no commission and no penalty. To our knowledge this is the first native bond market on a Monero-class privacy chain.

8.1 Transaction Types

Type	Purpose
`MARKET_ASK` (10)	List a deposit for sale, update the price, or delist (`ask_price = 0`)
`LOCK_COLLATERAL` (8)	Buyer immobilises the purchase amount in a collateral lock
`CANCEL_LOCK` (9)	Buyer voluntarily cancels the lock (funds return)
`TRANSFER_DEPOSIT` (7)	Deposit ownership transfer (also usable for direct OTC transfer)
`MARKET_PAYOUT_CLAIM` (11)	Seller claims the deferred sale proceeds

8.2 Protocol

1. **Ask**: the seller publishes an on-chain ask for a named `deposit_id` at a chosen price (above or below principal — the market decides).
2. **Collateral lock**: a buyer locks the purchase amount. Lock lifetime is consensus-bounded (`50 … 1 440 blocks`); every lock is indexed by expiry height.
3. **Atomic match**: when a valid lock matching an active ask is confirmed, consensus executes the exchange **in a single block**: the deposit's `owner_key` becomes the buyer's, and the sale proceeds are recorded as a deferred payout for the seller. All or nothing.
4. **Seller payout**: the seller claims proceeds with `MARKET_PAYOUT_CLAIM` (public, deposit-referenced; Dilithium co-signature mandatory from HF v23, §10.2).
5. **Expiry / cancellation**: unmatched locks expire automatically at their indexed height and the collateral is released; the buyer may also cancel voluntarily at any time before a match.

8.3 Safety Properties

- Mempool conflict rules prevent racing operations on the same deposit (e.g. a `TRANSFER_DEPOSIT` cannot coexist with a pending `INHERIT_REQUEST` ; claim/withdraw/transfer combinations on one deposit are serialised).
- Every step (ask, lock, match, payout, expiry) writes an undo record; chain reorganisations restore the exact prior state.
- The buyer's lock, like a deposit's principal, is consensus-immobilised — no counterparty risk, no custodian.

---

9. Trustless Inheritance ("Dead Man's Switch")

A consensus-native succession mechanism on a Monero-class privacy chain. Identities stay protected: the principal acts behind ring signatures, the heir is designated by a stealth-capable address; observers can see that an inheritance record exists and its amounts, never who is involved.

9.1 Roles and On-Chain Objects

Operation	TX type	Signed by	Effect
`INHERIT_SETUP` (4)	principal	Registers/updates the heir (spend+view pubkeys) in an LMDB record keyed by the principal's spend key; resets the activity clock	
`INHERIT_REQUEST` (5)	**designated heir only**	Opens a claim: `request_active = 1`, `request_height = h`; execution becomes due at `h + WAIT_BLOCKS`	
`INHERIT_CANCEL` (6)	principal	**Removes the entire designation** (record, pending request, stored testament). Re-arming requires a new SETUP	
`INHERIT_TESTAMENT` (13)	principal	On-chain carrier of the pre-signed testament (§9.3)	
`INHERIT_SWEEP` (12)	(pre-signed)	Marker carried by the testament's sweep transactions, enforced by the consensus gate (§9.5)	

Every operation's signature is verified by consensus against domain-separated messages (`PBC_INHERIT_*_V1` prefixes). A request from anyone other than the recorded heir is rejected.

9.2 The Waiting Period

```
PBC_INHERIT_WAIT_DAYS   = 540 days (18 months)
PBC_INHERIT_WAIT_BLOCKS = 540 × 86400 / 60 = 777 600 blocks
```

Execution of a request is permitted only when **both** hold at block height `H`:

```
H ≥ request_height + 777 600
last_activity_height ≤ request_height      (no principal activity since the request)
```

The waiting period is a single consensus constant; it is not user-configurable.

9.3 The Testament (liquid balance transfer)

Deposits can be reassigned by consensus (§9.6), but the principal's **liquid** RingCT balance cannot be spent by anyone else — so the principal's wallet maintains a *testament*: pre-signed sweep transactions to the heir.

- The wallet-RPC service auto-maintains it while synced: on first setup, and whenever the unlocked balance has grown by ≥ 100 PBC, it consolidates UTXOs, waits 3 blocks for confirmation, then rebuilds the sweeps.
- Each sweep carries two extra fields: the PBC type `INHERIT_SWEEP` and the principal's spend key — this marking is what the consensus gate (§9.5) enforces.
- The testament is stored twice: in the node's LMDB (daemon RPC, which verifies an active inheritance record exists), and **on-chain** in carrier transactions of type `INHERIT_TESTAMENT` signed by the principal over `H(prefix || principal || seq || hash(testament))`, with an extended tx-extra allowance (up to 40 960 bytes). On-chain storage means the testament survives even if every node that held it resynchronises.

9.4 Proof of Life

Only operations that **require the principal's spend key** reset the inactivity clock: `TERM_DEPOSIT`, `TERM_WITHDRAW`, `INHERIT_SETUP`. Permissionless operations (notably CLAIM, which any third party may submit against any deposit) deliberately do **not** count — otherwise an attacker could grief the heir by keeping a dead principal "alive" forever. Ordinary RingCT transfers are anonymous by design and therefore cannot (and must not) be attributed to the principal as activity.

9.5 Consensus Sweep Gate (premature-broadcast protection)

A pre-signed sweep is an ordinary valid transaction; without protection, anyone holding the blob could broadcast it early and bypass the waiting period. PBC closes this at consensus level (active since HF v21):

- `check_tx_inputs` — shared by mempool admission and block validation — rejects any transaction marked `INHERIT_SWEEP` unless the referenced principal's inheritance has been **executed**, and only within the window `[B, B + 2 880 blocks]` of the execution height `B`.
- The `executed[P] = B` flag is written during execution, *before* the sweeps are handed to the mempool, so legitimate sweeps pass; premature ones are rejected everywhere.
- Reorg undo restores `executed[P]`, the testament, and reassigned deposits exactly.

9.6 Execution

At each block, consensus scans active inheritance records; for every record that is due (§9.2) it executes in three passes:

1. **Collect** eligible records (read-only).
2. **Apply**: snapshot the testament; reassign every deposit whose `owner_key` is the principal's to the heir's spend key; write a comprehensive undo blob (previous record, affected deposit ids, testament snapshot); clear the request; set `executed[P] = B`.
3. **Broadcast**: parse the testament (bounded: 1–64 transactions) and submit each sweep to the mempool through the standard relay path — the gate admits them inside the execution window.

The heir's wallet then discovers inherited deposits through the daemon (deposits-by-owner lookup) and receives the swept liquid balance as ordinary incoming transfers. If no testament exists at execution time, deposits still transfer; only the liquid balance stays unswept (logged as a warning).

9.7 Observability

Daemon and wallet RPC expose the full state (heir set, request active, request height, blocks remaining, last activity, `wait_blocks`), so interfaces always display the countdown from live consensus values.

---

10. Hybrid Post-Quantum Cryptography

PBC layers NIST-standardised post-quantum primitives on top of the classical CryptoNote stack. Security holds as long as **either** layer resists.

10.1 PQC Registration

A wallet binds PQC keys to its spend key with a `PQC_REGISTER` transaction carrying the ML-DSA-65 public key (1 952 B), the ML-KEM-768 public key (1 184 B), and a proof of possession: an Ed25519 signature over `H("PBC_PQC_REGISTER_V1" || spend_pubkey || pqc_hash)`, verified by consensus. v2 addresses commit to the `pqc_hash` directly in the address encoding.

10.2 Post-Quantum Spend Authority (HF v23)

Withdrawals and market payouts authorise the spend of a *named, public* deposit (`deposit_id` and `owner_key` are on-chain), so they carry no ring anonymity to protect. Exactly there, PBC adds a genuine PQ authorisation requirement **without any anonymity loss**: alongside the Ed25519 owner signature, consensus requires an **ML-DSA-65 (Dilithium) co-signature** by the owner's registered PQC key over `H("PBC_PQC_WITHDRAW_V1" || deposit_id || payout_amount)`.

- Before HF v23: accepted if present (soft transition). From HF v23 (mainnet table: height 1 000): **mandatory** on `TERM_WITHDRAW` and `MARKET_PAYOUT_CLAIM`; rejected otherwise by mempool and block validation alike.
- Breaking Ed25519 alone no longer authorises a withdrawal; the attacker would also need to forge Dilithium, which Shor's algorithm does not break.

10.3 Hybrid Stealth Key Exchange

Stealth-output derivation for v2 addresses combines classical ECDH with an **ML-KEM-768 (Kyber)** encapsulation (ciphertext 1 088 B in tx-extra), merged through HKDF. Compromise of one KEM does not expose the other.

10.4 Honest Scope

Ordinary RingCT transfers remain classically protected: their spend authority is the CLSAG ring, whose anonymity model is incompatible with a naive per-output PQC signature. PBC's PQ guarantees therefore cover **registration, deposit spend authority, and stealth derivation** — the deposit-banking surface — and are hybrid everywhere they apply. (See the project's quantum analysis document for the full discussion.)

---

11. Determinism and Reorg Safety

- Integer-only arithmetic; all intermediate products in unsigned 128-bit; no platform-dependent rounding; fixed output-index rules.
- Every state-mutating PBC operation writes an explicit undo record: deposit tx undo, inheritance tx undo (with activity-only and testament-carrying variants), inheritance execution undo (record + deposit list + testament snapshot), collateral-lock and market undo, insurance-overflow deltas.
- `pop_block` is the strict inverse of block application for every subsystem; undo keys are fully cleared after use.
- Monotonicity of both global indexes is consensus-asserted; cross-node state hashes must match.

---

12. Security Considerations

Risk classes addressed by design and by remediation during the audit programme:

- Fee leakage / conservation violations (per-block assertions on every split).
- Reorg drift (systematic undo records; symmetric apply/pop for all PBC subsystems).
- Overflow mis-accounting (128-bit intermediates; explicit supply-base and destroyed-vs-supply checks).
- Commitment mismatch on public outputs (fixed index-0 rule, mask = 0, BP+ verification).
- Deposit-tier abuse via inflated unlock heights (consensus upper bound on `unlock_height`).
- Inheritance griefing via permissionless claims (proof-of-life restricted to spend-key operations).
- Premature testament broadcast (consensus sweep gate + execution window, HF v21).
- PQC key-binding forgery (registration proof-of-possession verified by consensus).
- LMDB corruption and buffer-safety classes (hardened record parsing; fixed-size packed records with versioning).

---

13. Parameter Summary

Parameter	Value
Max supply	18 446 744 PBC (no tail emission)
Emission	`(supply − emitted) >> 22` per block at 60 s
Block target	60 s (1 440 blocks/day)
Proof of Work	RandomX v2, `rx/pbc` variant (CPU-only)
Distribution period	1 440 blocks (~1 day)
Block split	Miner 91 % · Dev 2 % · Fee Pool 3.5 % · Deposit Pool 2.5 % · Insurance 1 %
Fee split	50 % miner · 50 % Fee Pool
Dynamic fee floor	×(1000 + locked_ratio)/1000, capped ×2
Miner vesting	4 × 25 % at ~1 / 30 / 60 / 90 days
Deposit tiers	30 / 90 / 180 / 270 / 365 days
Tier multipliers	1.0× / 1.3× / 1.6× / 1.8× / 2.0×
Min deposit / max per address	10 PBC / 10 deposits
Early-withdrawal penalty	2 % → Insurance Pool
Insurance cap	184 467 PBC (1 % of max supply); excess burned
Insurance subsidy	tops Deposit Pool up to 100 PBC floor, ≤ 10 % of insurance per period
LSM	dampening above 60 % locked ratio, floor 10 %
Market collateral lock lifetime	50 – 1 440 blocks
Inheritance waiting period	540 days = 777 600 blocks
Inheritance sweep window	2 880 blocks after execution
Testament auto-resign threshold	+100 PBC unlocked balance
PQC	ML-DSA-65 + ML-KEM-768 (hybrid); mandatory spend authority from HF v23

14. Conclusion

Privacy Bank Chain combines CryptoNote-grade privacy with native, consensus-enforced banking: real-yield term deposits with provable solvency, fee redistribution with a demand-linked fee floor, an insurance safety net that doubles as a deterministic deflation mechanism on a strictly finite supply, penalty-based early exit and a penalty-free atomic bond market, trustless inheritance protected by a consensus gate, and a hybrid post-quantum layer on every public spend-authority path — all in fixed, auditable C++, with strict integer determinism and full reorg symmetry.

Appendix A — PBC Transaction Types and Extra-Tag Space

PBC reserves tx-extra tags `0x50 … 0x65`. The PBC type of a transaction is the 1-byte payload of tag `0x50`.

Type	Name	Purpose
1	`TERM_DEPOSIT`	Create a tiered time-locked deposit
2	`CLAIM`	Extract accrued index rewards (permissionless)
3	`TERM_WITHDRAW`	Early exit with 2 % penalty (PQC co-signed from HF v23; CLI-only in v8.2.11 — see §7)
4	`INHERIT_SETUP`	Designate / update an heir
5	`INHERIT_REQUEST`	Heir opens the inheritance claim
6	`INHERIT_CANCEL`	Principal revokes the designation
7	`TRANSFER_DEPOSIT`	Deposit ownership transfer (market or OTC)
8	`LOCK_COLLATERAL`	Buyer-side purchase collateral
9	`CANCEL_LOCK`	Voluntary collateral release
10	`MARKET_ASK`	List / reprice / delist a deposit
11	`MARKET_PAYOUT_CLAIM`	Seller claims deferred proceeds (PQC co-signed from HF v23)
12	`INHERIT_SWEEP`	Marker on testament sweep TXs (consensus-gated)
13	`INHERIT_TESTAMENT`	On-chain testament carrier

Auxiliary tags carry: deposit / claim / withdraw payloads (`0x51–0x53` , `0x56`), ownership key and signature (`0x54–0x55`), inheritance payloads (`0x57–0x59` , `0x64–0x65`), market payloads (`0x5A–0x5E`), and the PQC material — Dilithium public key and signature, Kyber public key and ciphertext, registration commitment (`0x5F–0x63`).

END OF WHITEPAPER v1.8