JANZEER BLOCKCHAIN

Security & Threat Model

Trust assumptions · BFT safety · mitigations

Document v1.0 · Janzeer Network · 2026

Janzeer Security & Threat Model

Janzeer is a fixed-supply, **permissionless stake-free** BFT Layer-1 with a non-custodial HD-wallet scheme: the node verifies signatures but never generates, holds, or transmits user keys. This document consolidates Janzeer's security goals, trust assumptions, and threat model for security-minded engineers, auditors, and integrators. It explains what the protocol guarantees (single-block BFT finality, replay protection including cross-chain replay, overspend impossibility, fixed supply), what it relies on (an honest two-thirds quorum of the active validator set, NTP-synced clocks, and client-side key custody), and how each realistic attack is mitigated or, where a residual risk remains, names it plainly. Exact byte-level cryptographic and protocol details live in the Yellow Paper; this document references them rather than restating them.

1. Security goals & trust assumptions

What the protocol aims to guarantee

- **Safety (no two conflicting finalized blocks):** at most one block is finalized at each height, as long as fewer than one-third of the active set is Byzantine.
- **Single-block finality:** a block that reaches a COMMIT quorum is final immediately, with no confirmation-depth waiting period and no probabilistic reorg.
- **Replay protection:** every transaction is bound to a per-account nonce (part of both the signed preimage and the account state hash) and to the **network-id** in the signed preimage, so a transaction cannot be replayed on the same account or across a different Janzeer network (the mainnet, or a private test network).
- **Overspend impossibility:** an account balance can never go negative, on the live path or during sync.
- **Fixed supply conservation:** total supply is `MAX_SUPPLY = 100,000,000` (scale 8), pre-allocated at genesis and never minted; fees are split (recycled or paid to the producer), never burned and never created.
- **Authenticated networking:** only peers that complete a mutual ECDSA handshake are admitted.
- **Rule-set integrity:** the consensus-critical parameter set is pinned to a genesis chain-spec digest that peers exchange and verify at handshake, so a node cannot quietly change the rules and stay on the network.

What it relies on (trust assumptions)

- **Honest super-majority of the active set.** Liveness and safety hold only while strictly more than two-thirds of the *actual* active-set size `N` are honest and reachable — the BFT quorum is `bftQuorum(N) = floor(2N/3) + 1`, i.e. the classic "2f+1" when `N = 3f+1`. The active set is capped at `promoters-count = 21`.

- **Synchronized clocks.** Each node depends on NTP synchronization within `ntp-offset-threshold = 250 ms` (tightened for 15-s wall-clock slots). A node whose clock is not synchronized, or that is not fully synced, halts block production rather than risk producing on a bad clock.
- **Client-side key custody.** The security of any account rests entirely on the holder protecting their mnemonic / private key. The node has no recovery path: there are no key-generation or seed endpoints, and the node never sees a private key.
- **A live quorum exists.** Finality requires a live quorum of the active set; a single node (or any sub-quorum) cannot finalize blocks.

2. Cryptographic assurances

The node is a **verifier, not a custodian**. All key material is derived and held client-side; the node only validates signatures and never exposes key-generation or seed endpoints (see §1 of the REST surface — every transaction type is `POST`-verified, none is signed server-side).

Property	Value
Curve	secp256k1, compressed (33-byte) public keys
Signature scheme	ECDSA, deterministic per **RFC-6979** (HMAC-SHA256 `k`)
Malleability defense	**canonical low-S enforced** (high-S signatures are rejected)
Signature encoding	**DER**, then **Base64**; signed over the raw 32-byte digest (clients must use `prehash:false`)
Transaction / block / state hashing	**double-SHA256**, lowercase hex
Mnemonic → seed	PBKDF2-HMAC-SHA512, 2048 iterations, 64-byte output
Address	`"0x" + hex(keccak256(compressedPubKey)[0:20])` — Keccak-256 over the 33-byte compressed key, first 20 bytes; addresses are all-lowercase hex
Derivation path	`m/0/0/0`, all non-hardened

Deterministic RFC-6979 nonces remove the catastrophic private-key-leak risk of a bad random `k`, and low-S enforcement removes signature malleability as a transaction-identity attack surface. The address scheme is deliberately **not** Ethereum's (Janzeer takes the *first* 20 bytes of Keccak over the *compressed* key). Byte-for-byte parity between the node and the reference wallets is proven by committed vectors (`build/wallet-parity-vectors.json`); the canonical test mnemonic `abandon…about` derives `0x06e1c0fa9955a700876f8cb0acc7f13fba9fb8ba`. **Exact preimage byte layouts, derivation math, and constants are specified in the Yellow Paper.**

3. Consensus & BFT safety / liveness

Janzeer combines permissionless validator block production with two-phase (PREPARE → COMMIT) BFT finality (PBFT/Tendermint-style). There is no staking, delegation, or voting: validators are admitted by running a node and submitting a `PromoterTx` with a non-refundable deposit, and are elected by registration order.

- **Fault tolerance.** The protocol tolerates `f` Byzantine validators out of `N` active validators, where the quorum is `bftQuorum(N) = floor(2N/3) + 1`. This is "2f+1" when `N = 3f+1`.
- **Two-phase finality.** Each active validator casts a signed PREPARE on a valid pending block; at quorum PREPARE it fully validates and casts COMMIT; at quorum COMMIT the block is **final** — single-block finality, no confirmation depth. Votes are bundled into a **QuorumCertificate** gossiped once per slot.
- **Safety vs liveness.** *Safety* here means no two conflicting blocks both reach a COMMIT quorum, so the finalized chain never forks under the fault assumption. *Liveness* means honest validators keep producing and finalizing blocks; it requires a reachable quorum and synchronized clocks. Under network partition, Janzeer favors safety: a side without quorum simply does not finalize.
- **Equivocation is provable and slashable.** A validator is allowed exactly one vote per slot per stage (tracked by `SlotVoteTracker`). A conflicting second vote is provable evidence, packaged as a system `SlashingTx`. Reaching `equivocation-ban-threshold = 2` distinct proven offenses triggers a **permanent ban**; the reporter receives a `slash-reporter-bounty = 5` bounty from the treasury.
- **Downtime ejection.** A validator that stays silent on its slot is ejected from the active set at the next epoch boundary (a liveness-preserving, non-slashing penalty). A validator may also retire gracefully with an `ExitPromoterTx`, which removes it from the active set and slot schedule at the next epoch boundary; the registration deposit stays non-refundable.
- **Corroborated rollback.** Fork choice is longest-valid-chain with a single tip that only moves forward. A rollback that would discard committed state requires the **same 2f+1 quorum** of distinct authenticated validators to corroborate (`RollbackCorroborator`); it is not a general reorg engine, which removes single-node history-rewrite attacks.

Slots are deterministic round-robin over the epoch's frozen active list: active window `time-slot-duration = 12000 ms` plus intermission `time-slot-interval = 3000 ms` = a 15000 ms (15-second) full slot. An epoch is `epoch-height = 17000` main blocks (≈ 3 days at 15-s blocks); the active set is re-elected at each genesis (epoch) boundary. The active set is still-privileged anchor validators plus the longest-registered elected validators, filled up to `promoters-count = 21`, with `rotating-producer-seats = 5` seats rotating fairly (registration-ordered) through a `producer-candidate-pool-size = 100` standby bench.

- **Automatic anchor phase-out.** The three genesis anchor validators start privileged (always-in, downtime-immune) and shed that privilege **one per epoch** (`anchor-`

`demotions-per-epoch = 1`) once enough **proven independent** validators exist — non-anchors that produced in every one of the last `validator-proven-epochs = 2` epochs. Counting *proven/sustained* validators rather than raw registrations makes the mechanism un-weaponizable: a Sybil flood of fresh registrations cannot force demotions. The privileged set never drops below `safety-floor-validators = 4`, and if the independent set later collapses below the floor, dormant anchors automatically **re-engage** (anti-fragile). Demotion is not removal — a demoted anchor keeps validating on merit like any registrant. The privileged count is committed per epoch in each GenesisBlock (`activeAnchorCount`) so it is reconstructable on every node.

4. Transaction-level protections

- **Replay protection (per-account nonce + network-id).** Every transaction carries a `nonce` that is part of the signed preimage *and* the account state hash. The mempool permits consecutive nonces for pipelining (`expected = confirmedNonce + pendingCount`); at apply time the check is strict (`state.nonce == expected`, then increment). A captured transaction cannot be replayed because its nonce is already consumed. The signed preimage *also* begins with the `network-id` string (`consensus.network-id`, default `janzeer`, distinct per chain), so a transaction signed for one network cannot be replayed on another. System `RewardTx` is the only nonce-exempt type (nonce 0, fee 0, never user-submitted).
- **Overspend / negative-balance impossibility.** A balance can never go negative on either the live or sync path. The mempool subtracts already-pending spend before accepting a new transaction, so an attacker cannot queue multiple transactions that jointly exceed the actual confirmed balance.
- **Fee rules.** `MINIMUM_FEE = 0.01`; validator registration (`PromoterTx`) has an exact `3` fee plus a `2000` **non-refundable deposit** to the treasury bank (the primary Sybil guard; exiting does not return it). Fees are split `treasuryFeeShareBps = 2000` (20% to the treasury bank, 80% to the producer) — never burned, never minted.
- **Mempool actual-balance check.** Admission validates against the sender's real balance (confirmed minus already-pending), not an optimistic estimate.
- **Deterministic ordering.** Within a block (capacity `block-capacity = 100`), the `RewardTx` applies first, then user transactions in ascending nonce — a deterministic order every node reproduces, so state hashes match across the network.

5. Network security

- **Authenticated mutual handshake.** Domain-separated `JANZEER_HANDSHAKE_V1` (folded with the `network-id`): Greeting(nonce + **chain-spec digest**) → Challenge(server nonce + proof) → Proof(sign server nonce). Both sides verify ECDSA signatures over `doubleSha256(DOMAIN || ":" || networkId || nonce)`; the node UID is the node's public key. **Only authenticated peers are added.**

- **Rule-set integrity (chain-spec pinning).** The consensus-critical parameter set (everything under `consensus.*`) is the canonical rule-set fixed at genesis. Each node computes a **chain-spec digest** (double-SHA256 over those params) and exchanges it in the Greeting; a peer whose digest differs is rejected before the challenge. A node that edits any consensus parameter forks itself off the network instead of silently diverging. Operational settings (ports, DB, peers, NTP) stay per-node.
- **Protocol-version gating.** Transport is Netty NIO TCP with a custom binary frame (`protocol-version string ‖ timestamp(long) ‖ 1-byte MessageType ‖ body`). The P2P protocol version is `2.0.0`; a version mismatch drops the frame, preventing cross-version confusion.
- **Message expiry.** Non-block messages older than `expiry = 10000 ms` are dropped, limiting replay/delay of stale gossip.
- **Peer limits.** Target inbound `peers-number = 3`, accept cap `allowedConnections = 7`, heartbeat 10000 ms — bounding resource exhaustion and connection-flood pressure.
- **Sync gating.** Block production is gated on `SYNCHRONIZED` status; a node that is behind or clock-skewed does not produce.

6. Economic security

- **Fixed supply, no inflation surface.** `MAX_SUPPLY = 100,000,000` is pre-allocated at genesis and never minted, so there is no minting code path to exploit for inflation. Block rewards are paid *out of* a pre-funded treasury, not created.
- **Key-less treasury.** The treasury "bank" account is `0x0000000000000000000000000000000000000000` — a capture-proof address with no corresponding private key. No one can sign a transaction from it; only protocol reward/fee logic moves its funds. It is funded `77,500,000` at genesis; `circulatingSupply = MAX_SUPPLY − bankBalance`.
- **Fee recycling conserves supply.** The emission model is "C + D": a declining scheduled subsidy `scheduledReward(h) = max(rewardTailFloor, rewardBlock / 2^(h / rewardHalvingPeriod))` (`rewardBlock = 10`, `rewardHalvingPeriod = 2,100,000` blocks ≈ 1 year at 15-s blocks, `rewardTailFloor = 0.5`), capped at `min(scheduledReward(h), bankBalance)`, plus treasury fee recycling (the 20% fee share that flows back to the bank). Net inflation is zero; supply is conserved by construction.
- **Reward validated on every node.** The `RewardTx` is system-generated (fee 0, nonce 0) and validated against the canonical computed value on every node; a block whose reward does not match the canonical amount is rejected. A producer cannot over-pay itself.
- **Reward goes 100% to the producer.** The full block reward is paid to the producing validator's wallet — there is no delegation, commission split, or reward claiming. Rewards pay for the *work* of running the chain, not yield on stake, and every node recomputes the same canonical amount.

7. Threat model

Threat	Vector	Janzeer mitigation / residual risk
Double-spend	Spend the same balance twice	Per-account nonce + mempool actual-balance check + single-block BFT finality: once a block is COMMIT-finalized it cannot be reorged without a 2f+1 corroborated rollback.
Replay (same chain)	Rebroadcast a captured signed tx	Nonce is in the signed preimage and state hash; the consumed nonce makes the replay invalid.
Cross-chain replay	Replay a tx on another Janzeer network	The `network-id` is bound into the signed preimage; a tx signed for one network (the mainnet, or a private test network) is invalid on any other.
Equivocation / double-vote	A validator votes twice in a slot/stage	`SlotVoteTracker` makes the second vote a provable `SlashingTx`; `equivocation-ban-threshold = 2` offenses → permanent ban; reporter bounty 5.
Long-range / history rewrite / rollback	Rewrite finalized history	Single forward-only tip; discarding committed state needs the **same 2f+1 quorum** of distinct authenticated validators (`RollbackCorroborator`). A lone or sub-quorum node cannot rewrite history.
Sybil peers	Flood the network with fake peers	Mutual authenticated handshake (node UID = public key); only authenticated peers admitted; accept cap `allowedConnections = 7`. Peer count confers no consensus weight — block production comes only from registered validators.
Sybil validators	Register many validators to capture the set	Each `PromoterTx` costs a `2000` **non-refundable deposit** (exiting does not return it) and one seat per key; election is by registration order with fair rotation. Anchor phase-out counts only *proven/sustained* validators (produced every one of the last `validator-proven-epochs = 2` epochs), never below `safety-floor-validators = 4`, so a flood of fresh registrations cannot weaponize demotions. **Residual risk:** the deposit is honestly softer than stake/PoW, sized for a small operator set.
Rule-set tampering	Quietly edit a consensus param	Consensus params are pinned to a genesis **chain-spec digest** exchanged and verified at handshake; a node with a different digest is rejected and forks itself off rather than silently diverging.
Eclipse	Isolate a node behind attacker-controlled peers	Dials shuffled known peers plus configured root/seed nodes until target peer count; sync gated on `SYNCHRONIZED`. **Residual risk:** as with any P2P network, an operator must configure trustworthy seed/root nodes; diverse, trusted seeds reduce eclipse exposure.

Threat	Vector	Janzeer mitigation / residual risk
Key theft	Steal a user's mnemonic / private key	Non-custodial design limits blast radius to the compromised account; the node never holds keys, so a node breach exposes no user keys. **Residual risk:** key custody is the holder's responsibility; there is no protocol-level recovery.
Clock manipulation	Skew a node's clock to forge slot timing	NTP sync required within `ntp-offset-threshold = 250 ms`; a node halts production when not synchronized. **Assumption:** honest NTP sources; slot ownership is deterministic, so a single skewed clock cannot steal another's slot, only forfeit its own.
Treasury drain	Spend the genesis treasury	The bank `0x000…0` is key-less and unspendable by signature; only protocol reward/fee logic moves it, and reward amounts are canonically validated per node.
Inflation bug	Mint coins beyond max supply	Fixed supply with no minting path; rewards paid from a finite pre-funded bank; reward validated on every node.
Signature malleability	Mutate a signature to alter tx identity	Canonical low-S enforced; deterministic RFC-6979 nonces.
Spam / resource exhaustion	Flood mempool or connections	Minimum fees (`MINIMUM_FEE = 0.01`; validator registration costs a `3` fee plus a `2000` non-refundable deposit), message expiry `10000 ms`, peer/connection caps, block capacity `100`.

8. Honest limitations & operational caveats

- **Dev/testnet committed secrets must be rotated.** The repository ships dev-default genesis anchor wallets and signer material for local and private test-network bring-up. These are **not production secrets** — operators must generate and configure fresh keys (via `janzeer.genesis.anchors` / `janzeer.genesis.signer-secret`) before any mainnet deployment.
- **Dev-fund faucet must be OFF in production.** The `janzeer.dev-fund` faucet (`0xaddr:amount`) is for development end-to-end testing and is OFF by default. It must never be enabled on a production network.
- **The registration deposit is non-refundable by design.** The `2000` `PromoterTx` deposit goes to the treasury bank and is never returned — not on graceful exit, not on ban. This is intentional (it is the primary Sybil guard); operators should treat it as a sunk cost, not a bond.
- **The 12M ecosystem pool's custody must be transparent.** The genesis ecosystem/distribution pool (`12,000,000`) should be held under a transparent, accountable mechanism (faucet / airdrop / governed distribution), not a private founder wallet, so its movements are auditable.

- **A single node cannot finalize.** Finality requires a live BFT quorum of the active set. A single node (or any group below `bftQuorum(N)`) can build and serve blocks but cannot finalize them; production also halts when not `SYNCHRONIZED` or clock-synced.
- **NTP dependency.** Safety of slot timing and production gating depends on NTP synchronization within `ntp-offset-threshold = 250 ms`. Operators should run reliable, ideally diverse, NTP sources.
- **This is not a formal audit.** This document describes the protocol's intended security properties as implemented in node version 0.1.0; it is not a proof of those properties and has not been independently audited. We recommend an independent security review and formal analysis before relying on Janzeer for high-value production use.

Document status

Version 1.0 — reflects node version 0.1.0. Exact byte-level cryptographic and protocol facts (preimage layouts, derivation math, hashing, consensus constants) are authoritative in the Yellow Paper.