ZKChan — Privacy-First Digital Money for the Modular, Cross-Chain Web3

Version: 0.1 (Draft)
Date: November 12, 2025
Project: ZKChan ("zkChan")

> *zkChan is a next-generation privacy coin that blends zero-knowledge cryptography with fast, scalable, and interoperable infrastructure. Inspired by Zcash's vision of confidential transactions, zkChan extends "privacy as a right" into a modular, cross-chain environment, enabling secure transactions without sacrificing transparency or compliance.*

Executive Summary

Public blockchains expose balances, counterparties, and transactional patterns by default. zkChan restores financial privacy as a default while preserving on-chain verifiability and practical compliance. Using zk-SNARKs for confidential transfers, a modular scalability stack, and a cross-chain privacy bridge, zkChan enables:

- **Private payments** with cryptographic integrity (no amounts, addresses, or metadata leaked).
- **Cross-chain settlement** via a burn-and-mint style privacy bridge that preserves unlinkability across ecosystems.
- **Selective disclosure** ("auditable privacy") enabling users or businesses to share verifiable views with auditors and counterparties when needed.
- **Developer-ready integrations** via SDKs and APIs for wallets, dApps, and institutional systems.

The result is a trust-minimized, censorship-resistant, and regulation-aware monetary primitive suited to both individuals and institutions.

---

1. Design Goals

1. **Default Confidentiality:** Private by default; transparency only by explicit consent.
2. **On-Chain Verifiability:** Every private transfer is fully validated on-chain via succinct proofs.
3. **Modularity & Performance:** Pluggable data availability and execution layers; predictable, low fees.
4. **Interoperability:** Native mechanisms to move private value across chains without deanonymizing users.
5. **Compliance-Enabling:** View/audit keys and selective disclosure proofs for real-world requirements.
6. **Open Governance:** Community-owned treasury and protocol evolution via zkChan DAO.

---

2. System Overview

zkChan is composed of the following core components:

- **Shielded Pool (SP):** A set of smart contracts (or an appchain module) that maintains a Merkle tree of **commitments** representing encrypted notes (ownership claims to value). Each spend consumes **nullifiers** proving one-time use without revealing which note.
- **Proof System:** zk-SNARK circuits validate: (a) knowledge of a note's spending key, (b) non-negative amounts with balance conservation, (c) membership of notes in the Merkle tree, and (d) sound forging of new commitments.
- **Address Model:** Distinct key roles for **spend**, **view**, and optional **audit** authorities. Addresses can be *transparent* (for bridging/UX) or *shielded* (private).
- **Cross-Chain Privacy Bridge (CPB):** A burn-and-mint style mechanism enabling private transfers across chains. Messages carry proof-attested state, preserving unlinkability.
- **Scalability Layer:** zkChan instances can deploy as an L2 validity rollup or as contracts on L1/L2s; data availability (DA) may be on the host chain or external DA networks.
- **Governance & Treasury:** Parameter upgrades, circuit migrations, and ecosystem grants are steered by zkChan DAO.

---

3. Cryptographic Foundations

3.1 Commitments, Nullifiers, and Notes

- **Note:** Encapsulates value and ownership metadata (e.g., value, asset id, recipient public key, blinding).
- **Commitment (C):** zk-friendly hash of a note; inserted into a Merkle tree.
- **Nullifier (N):** Derived from the note and spending key to prevent double-spends; published when a note is consumed.
- **Merkle Root (R):** Represents the current state of all commitments; proofs refer to a recent valid root.

3.2 zk-SNARKs

- zkChan uses a modern **plonkish** proving system (e.g., PLONK/UltraPlonk) with a universal or updatable setup to avoid single-use trusted ceremonies.
- Circuits enforce value conservation, valid key ownership, note inclusion, and correct transition to new commitments.
- Hashes are **SNARK-friendly** (e.g., Poseidon/Rescue) for efficient in-circuit computation. *(Final choices documented in the Reference Implementation.)*

3.3 Selective Disclosure

- **View Keys:** Permit decryption of incoming/outgoing notes for a specific account without spend authority.
- **Audit Keys:** Optional scoped keys enabling third parties (auditors, compliance teams) to reconstruct transaction history or generate attestations without revealing counterparties beyond scope.
- **Proofs of Compliance:** Users can generate zero-knowledge attestations (e.g., solvency, source-of-funds within a period) without exposing raw transaction data.

---

4. Transaction Lifecycle

1. **Deposit (T→Z):** Funds move from a transparent source into the **Shielded Pool**, creating new commitments and updating the Merkle root.
2. **Shielded Transfer (Z→Z):** Sender proves ownership and balance conservation, consumes prior notes (nullifiers) and creates fresh commitments for recipients. No addresses, amounts, or linkages are revealed.
3. **Withdrawal (Z→T):** Recipient reveals a public output to exit the pool, with proofs ensuring no leakage of the shielded graph.
4. **Cross-Chain Transfer (Z@A→Z@B):** Via CPB, a user privately “burns” (consumes) value on Chain A and “mints” (creates commitments) on Chain B using a message+proof that links states without revealing identities.

---

5. Cross-Chain Privacy Bridge (CPB)

5.1 Objectives

- Preserve **unlinkability** across chains (no shared identifiers).
- Provide **finality-aligned** settlement (respecting L1/L2 reorg assumptions).
- Minimize **trusted relayer assumptions** via on-chain verification of zk proofs and light-client checks where feasible.

5.2 Flow (Conceptual)

1. **Private Burn on Source (A):** User creates a shielded transfer proving consumption of notes (nullifiers published) and generating a CPB **bridge message** M containing: amount, asset id, destination chain id, and a commitment to the recipient’s target-chain note—all inside a zk proof Π_A.
2. **Attestation & Relay:** Validators/relayers observe finalized state on A and forward (M, Π_A) to the destination chain B.
3. **Mint on Destination (B):** Contract on B verifies Π_A and light-client/finality checks for A, then inserts the target commitments into B’s shielded pool and emits receipts.
4. **Spending on B:** Recipient now holds spendable private notes on B.

   Note: Exact finality and light-client mechanisms vary by pair (e.g., Ethereum↔L2, Cosmos IBC-style, Solana↔EVM). The protocol is modular to swap in chain-specific verification gadgets.

5.3 Security Considerations

- **Replay Protection:** Nonces/epochs bound to chain ids; nullifiers unique per domain.
- **Double-Mint Prevention:** Destination verifies that source nullifiers are finalized and globally unique.
- **Liveness:** Multiple permissionless relayers; fees and MEV resistance via randomized message encoding.

---

6. Scalability & Architecture

- **Deployment Modes:**
- **Smart-Contract Mode:** zkChan as contracts on EVM/L2/SVM environments.
- **App-Rollup Mode:** zkChan as a validity rollup with its own sequencer set and DA (e.g., on-host DA, Celestia, EigenDA).
- **Batching & Aggregation:** Rollup batches multiple shielded transfers, amortizing proof costs and gas.
- **Prover Networks:** Optional distributed provers (market of proving services) to offload heavy computation from end-users.
- **Upgradability:** Governance-gated circuit and parameter upgrades with timelocks and emergency pause only for non-custodial risk mitigation.

---

7. Address & Key Model

- **Spend Key:** Authorizes spending of notes.
- **View Key:** Allows decryption/visibility for accounting and wallet UX.
- **Audit Key (optional):** Time-scoped, revocable disclosure to auditors/partners.
- **Address Types:**
- **Shielded Address (z-addr):** Default private address format derived from spend/view keys.
- **Transparent Address (t-addr):** For interoperability, bridging, listing fees, or UX-critical flows.
- **Rotation:** Users can rotate or split view/audit keys without affecting spend keys.

---

8. Economic Model (Draft)

Final token parameters are subject to community decision; below is a reference design.

- **Fee Asset:** Fees are payable in the host chain’s native gas token or in **ZKC** (placeholder ticker) depending on deployment mode.
- **Protocol Fees:** A small percentage of transaction fees routes to the **zkChan Treasury** for R&D, grants, audits, and relayer incentives.
- **Relayer Incentives:** CPB relayers earn fees for message delivery; competitive market encourages low latency.
- **Treasury Policy:** DAO-controlled, with programmatic budgeting and milestone-based disbursements.

---

9. Governance (zkChan DAO)

- **Scope:** Parameter updates (e.g., fee rates, circuit versions), bridge route listings, grant approvals.
- **Mechanism:** On-chain proposals, quorum thresholds, and timelocked execution.
- **Checks & Balances:** Multi-sig guardianship for emergency circuit halts (non-custodial), bounded by DAO oversight.
- **Transparency:** All governance artifacts and audits published publicly; selective disclosure proofs for sensitive data (e.g., grant recipient compliance attestations).

---

10. Developer Platform

10.1 SDKs & APIs

- **Wallet SDK:** Key management (spend/view/audit), note encryption/decryption, proof orchestration.
- **dApp SDK:** High-level functions: `deposit()`, `transferShielded()`, `withdraw()`, `bridgeTo(chainId)`, `proveDisclosure(scope)`.
- **Bridge SDK:** Tools for relayers: chain light-client adapters, message queues, and receipt monitoring.

Example (pseudocode):

```
const w = Wallet.fromMnemonic(m);
const r = await sdk.transferShielded({
  to: recipientZaddr,
  amount: "100.0",
  assetId: "ZKC",
  feeHint: "auto",
});
await sdk.submit(r.tx, r.proof);
```

10.2 Integrations

- **Exchanges & Custodians:** View/audit key workflows for regulated reporting.
- **Commerce:** Payment requests with one-time view scopes and refundable escrow notes.
- **DeFi:** Private liquidity provision and swaps via shielded AMMs; cross-chain settlement via CPB.

---

11. Privacy, Threat Model & Limitations

- **Network Metadata:** zk proofs hide ledger data but not IP-layer metadata; recommend Tor/relays/Privacy-Enhanced RPCs.
- **Linkability Risks:** Pattern analysis (timing/amount correlations) mitigated via decoy outputs, randomized fees, and output padding.
- **Trusted Setup:** If a setup is required, use updatable ceremonies and public transcripts; prefer universal setups where possible.
- **Circuit Security:** Independent audits, formal verification where feasible, and continuous fuzzing of cryptographic primitives.
- **Bridge Risk:** Chain reorgs and light-client assumptions are explicit; economic bounds and confirmation windows enforced.

---

12. Compliance-Enabling Features

- **Selective View:** Grant auditors access to a time-bounded ledger slice via audit keys.
- **ZK Attestations:** Proofs of solvency, spend-within-limits, or jurisdictional compliance without raw data exposure.

- **Revocation & Rotation:** Revoke audit scopes at any time; new keys do not affect on-chain funds.
- **Data Retention:** Off-chain encrypted note archives for users/institutions with configurable policies.

---

13. Roadmap (Indicative)

- **Phase 0 — Research & Specs:** Finalize circuits, key formats, and CPB design; begin audits.
- **Phase 1 — Testnet (Contract Mode):** Shielded pool, deposits/withdrawals, basic Z→Z transfers, wallet SDK alpha.
- **Phase 2 — CPB Testnet:** Privacy bridge between two target chains; relayer market; light-client adapters.
- **Phase 3 — App-Rollup Beta:** Validity rollup with external DA; batch proving; fee market experiments.
- **Phase 4 — Mainnet:** DAO genesis, treasury live, audits published, compliance SDKs GA.

---

14. Implementation Notes (to be finalized)

- **Hash Function:** Poseidon (tbd parameters).
- **Curve & Proof System:** Plonkish over a pairing-friendly curve (e.g., BLS12-381) with universal setup.
- **Address Encoding:** Bech32-style with network prefix; QR-safe.
- **DA Options:** Host-chain calldata vs. external DA (Celestia/EigenDA) — configurable per deployment.
- **Wallet Formats:** BIP-style mnemonic derivations for spend/view keys with hardened paths.

---

15. Glossary

- **Commitment:** Hash binding to hidden note data.
- **Nullifier:** One-time identifier preventing double-spends without revealing the spent note.
- **Shielded Pool:** Contract/module holding commitments and verifying zk proofs.
- **Selective Disclosure:** Sharing verifiable facts without revealing underlying data.
- **DA (Data Availability):** Mechanism ensuring transaction data is persistently accessible.

---

16. Disclaimers

This document is for informational purposes only and does not constitute investment, legal, or tax advice. Cryptographic systems are complex and carry risk. Specifications herein are subject to change through community governance and further security review.

---

17. Core Features (Recap)

- **zk-SNARK Shielding:** Fully confidential transactions secured by zero-knowledge proofs.
- **Cross-Chain Privacy Bridge:** Private transfers and swaps across multiple blockchains.

• **Scalable Layer Design:** Modular architecture enabling low fees and high throughput.
• **Auditable Privacy:** Selective disclosure for compliance or verification.
• **Community-Governed Treasury:** Decentralized funding & governance via zkChan DAO.
• **Developer-Ready SDKs:** Tooling for integrating private payments and zk-proof logic into dApps.

Appendix A — Example Data Structures (Illustrative)

```
Note {
  assetId: bytes32,
  value: u128,
  pkRecipient: bytes32,
  rho: bytes32,       // note randomness
  r:   bytes32        // blinding
}

Commitment C = Poseidon(assetId || value || pkRecipient || rho || r)
Nullifier  N = Poseidon(skSpend || rho) // domain-separated per chain
```

Appendix B — Example dApp Flows (High Level)

1. **Pay:** Generate shielded tx → produce proof → submit → recipient scans → note spendable.
2. **Swap (Private AMM):** Deposit → proof-gated swap inside pool → withdraw shielded.
3. **Bridge:** Z→Z across chains via CPB message and verification on destination.

For questions or collaboration, include links to your repositories, docs site, and community channels here (TBD by project owners).