A Compliant Privacy Protocol on Solana

A Next-Generation Wallet and Token Ecosystem Built for Speed, Security, and Institutional Confidence

WhitePaper V2.0 - February 2026

Email: privacy@noc-tura.io
Website: www.noc-tura.io

1. Introduction

1.1 Vision & Mission

Noctura is built on a fundamental principle: **privacy is a human right, not a privilege**. In an era where every blockchain transaction is permanently recorded and publicly traceable, financial privacy has become a critical vulnerability rather than a protected freedom.

Our vision is to establish a new paradigm for blockchain privacy—one that doesn't force users to choose between confidentiality and compliance, between security and usability, or between privacy and participation in the mainstream financial ecosystem.

Noctura delivers the **first shielded privacy layer on Solana**, combining institutional-grade cryptography with a user-friendly dual-mode wallet that makes privacy accessible to everyone—from individual users protecting their financial dignity to enterprises managing confidential treasury operations.

We believe the future of blockchain finance must include privacy as a foundational feature, not an afterthought. Noctura makes this future possible today.

Design Goals

- **Usability first:** One-tap privacy with clear prompts and safe defaults
- **Private DeFi access:** Shielded swaps enabling confidential trading without revealing positions or strategies
- **Provable correctness:** Zero-knowledge verification anchored on-chain
- **Compliance readiness:** Selective disclosure when required, not blanket surveillance
- **Realistic performance:** Conservative throughput targets and staged optimizations (batching, aggregation, GPU proving)
- **Security by process:** Audits, formal review of circuits, bug bounties, incident response

1.2 What Noctura Is (Compliance-First Privacy Infrastructure + Dual-Mode Wallet)

Shielded Privacy Layer on Solana

A protocol that maintains commitments, nullifiers, and Merkle roots on Solana. Proofs are generated off-chain and verified on-chain by lightweight programs that update shielded state. This is a privacy overlay, not an L2 rollup.

Dual-Mode Wallet (primary interface)

- **Transparent Mode (public):** Behaves like a standard Solana wallet for full DeFi/NFT compatibility
- **Shielded Mode (private):** Sender, receiver, and amount are hidden; transfers update shielded state via zk-proofs
- **Shielded Swaps:** Private token exchanges within the wallet without revealing trading activity, amounts, or counterparties on-chain

- **Cross-Mode transfers:** Public ⇄ shielded with unlinkability guarantees
- **Selective disclosure:** View Keys (scoped read access) and Audit Tokens (consent-bound assertions) let users prove legitimacy to exchanges/partners without exposing their full history
- **Developer SDKs:** Simple methods to send shielded transactions, execute private swaps, request disclosures, and integrate compliance prompts

Performance Posture (realistic)

We target hundreds of shielded TPS at launch, scaling via batching, proof aggregation, and GPU provers. zk-proof generation is computationally intensive; on-chain verification and Solana runtime limits make "65,000 TPS shielded" unrealistic and we do not claim it.

1.3 Key Differentiators & Compliance-Ready Design

1) Wallet-first UX

Privacy is a toggle, not a new chain. Clear threshold prompts, geo-fencing where required, and educational tooltips reduce user error.

2) ZK-native unlinkability (no mixer dependency)

Properly constructed shielded transfers already break linkability. We avoid "advanced mixing" as a core dependency; any future mixer-style batching would be optional and scoped to UX/fee behavior, not anonymity.

3) Proof system pragmatism

A SNARK-first design provides succinct proofs with fast verification. STARKs are an optional roadmap module (e.g., for large batch audits or post-quantum posture) and will only be introduced when there is a clear, measured benefit.

4) Selective disclosure done right

- **View Keys:** Scoped (tx, time window, balance range), revocable
- **Audit Tokens:** Consent-bound, expiring credentials that validate facts (KYC pointer, origin proof) without revealing raw history

5) Compliance survival, not evasion

- Transparent-by-default onboarding; private mode is opt-in
- KYC/Travel-Rule adapters for partners; restricted jurisdictions enforced via geo-fencing/KYC country
- Team KYC and contract audits are published before listings
- On-chain presale with public contract address for verifiability

6) Security process, not slogans

- Third-party audits of presale/token/verification programs
- Cryptography reviews of circuits and parameters
- Bug bounty and runtime anomaly detection

- Documented incident response and upgrade path via guarded governance

7) Sustainable economics

- Token used for shielded fees and prover/relayer incentives
- Staking tiers post-TGE (APR 128%/68%/34% for 365/182/90-day locks)
- DAO-governed parameters, with optional fee-burn to align long-term value

Result: Noctura is shielded privacy layer on Solana with a dual-mode wallet that makes private, compliant transactions practical, focusing on usable privacy, measured performance, and verifiable security.

2. Project Overview

2.1 Mission & Principles (Privacy with Legitimacy)

Mission: Make private transactions on Solana usable, verifiable, and compliant—so individuals and institutions can choose confidentiality without sacrificing speed, security, or market access.

Principles:

- **User choice:** Transparent by default; opt-in shielded mode with clear consent
- **Provable correctness:** Zero-knowledge proofs verified on-chain; no "trust me" privacy
- **Selective disclosure:** Fine-grained View Keys and Audit Tokens for lawful proofs without bulk deanonymization
- **Pragmatism > hype:** Realistic throughput, staged optimizations, measured feature rollout
- **Security as a process:** Audits, formal reviews, bug bounty, incident response, and guarded governance
- **Compliance survival:** Geo-fencing, threshold prompts, KYC/Travel-Rule adapters; published legal/audit artifacts

2.2 Positioning on Solana (Privacy Overlay, not a rollup)

What it is: A shielded privacy overlay that maintains commitments, nullifiers, and Merkle roots on Solana. Proofs are generated off-chain and verified by Solana programs that update shielded state. Includes shielded swap aggregation layer for private token exchanges within the privacy overlay.

What it is not:

- Not a sidechain, bridge, or EVM-style rollup
- Not a mixer-only system; unlinkability derives from robust ZK design

Why this fits Solana:

- Uses Solana for finality, availability, and composability, while moving heavy proving off-chain
- Targets hundreds of shielded TPS initially; scales via batching, aggregation, and GPU provers—not "65k TPS" claims

- Integrates with existing DeFi/NFT apps via a lightweight Wallet SDK and Verifier Program interfaces

2.3 Users & Use Cases (Individuals, Traders, Enterprises)

Individuals (privacy & safety)

- **Private payments & remittances:** Hide salary, gifts, and recurring payments from public graphs
- **Private DeFi access:** Swap tokens privately for everyday use without exposing trading patterns
- **Portfolio privacy:** Exchange assets without revealing holdings or investment strategies
- **Financial safety:** Reduce doxxing, extortion risk; keep balances and counterparties confidential
- **Selective proofs:** Provide scoped proofs of funds to ramps/exchanges without revealing history

Traders & Funds (strategy protection)

- **Private swaps:** Execute token exchanges without revealing positions, preventing front-running and copy-trading
- **MEV protection:** Swap tokens privately to avoid sandwich attacks and value extraction
- **Strategy confidentiality:** Rebalance portfolios without broadcasting intentions to competitors
- **Front-run resistance:** Execute swaps, reallocations, and rebalances privately to avoid copy-trading and MEV-style inference
- **Treasury moves:** Move inventory and collateral without telegraphing intent
- **Audit on demand:** Prove asset origin/solvency to counterparties via Audit Tokens

Enterprises & Institutions (confidential operations)

- **Payroll & vendor settlement:** Confidential flows with verifiable compliance
- **Treasury & M&A:** Conceal strategic movements; disclose only to auditors/regulators as required
- **Policy controls:** Scoped view access for finance, compliance, and external auditors; revocation and expiry baked in

Developer & ecosystem integrations

- **Wallet SDK hooks:** `sendShieldedTx`, `executeShieldedSwap`, `crossModeTransfer`, `generateViewKey`, `issueAuditToken`
- **dApp patterns:** Private swaps, shielded lending/repayment, NFT settlement with selective provenance
- **Compliance adapters:** Travel-Rule providers, KYC hash-pointers, and geo-rules enforced at wallet boundary

Outcome: Noctura delivers usable, compliant privacy for everyday users, competitive traders, and regulated enterprises, on Solana, with a dual-mode wallet that makes switching between public and private flows trivial.

3. Noctura Privacy Wallet

3.1 Dual-Mode UX: Transparent (Public) vs Shielded (Private)

- **Transparent Mode (default):** Standard Solana account (Ed25519). Full compatibility with DeFi/NFT apps; transactions are public on-chain
- **Shielded Mode (opt-in):** Wallet derives shielded keys and maintains private balances in the shielded state (commitments/nullifiers/Merkle root). Sender, receiver, and amount are hidden; validity is enforced by zk-proofs
- **Shielded Swaps (private exchange):** Within Shielded Mode, users can swap tokens privately without revealing:
    - Trading pairs
    - Swap amounts
    - Wallet balances
    - Transaction timing

    All swaps are executed via zk-proofs with amounts and counterparties hidden
- **One-tap toggle:** The wallet surfaces a clear Public ⇄ Private switch with legal/geo prompts where required. Mode and fee estimates are shown prior to signing
- **Predictable fees:** UX shows (i) shielded fee in $NOC (converted from USD), (ii) expected confirmation window (batch/aggregation latency), and (iii) disclosure requirements if thresholds are crossed

3.2 Cross-Mode Transfers (Public ⇄ Shielded) with Unlinkability

Public → Shielded (Deposit):

1. User funds a deposit program
2. Wallet creates a commitment `C = Commit(sk, amt, r)` and submits zk-proof of well-formedness
3. On-chain verifier updates Merkle root
4. Shielded note appears in private balance

Linkability: Broken at the proof boundary—no public link to the resulting note.

Shielded → Public (Withdraw):

1. Wallet spends a note using a nullifier N
2. zk-proof shows note membership + non-spent
3. Verifier marks N and releases public funds to a chosen address

Unlinkability: Exit address is not linkable to the original deposit or prior shielded hops.

Shielded → Shielded (Transfer):

Multi-output spend with change; zk-proof updates commitments and nullifiers atomically. No address/amount reveal.

Best practices: Randomized timing, output aliasing, and batch joins to strengthen anonymity sets.

3.3 Selective Disclosure: View Keys & Audit Tokens (Scopes, TTL, Revocation)

View Keys (read-only)

- **Scope:** Single transaction, time window, balance range, or proof-of-funds
- **Derivation:** From user's disclosure branch key; never grants spend authority
- **Revocation:** Local list + on-chain revocation registry (hash pointer) for partners to honor

Audit Tokens (consent-bound assertions)

- **Payload:** `{auditor_pubkey, scope, ttl, zk_assertions[], nonce, sig_user}`
- **Use cases:** KYC proof-of-origin, solvency attestations, Travel-Rule data exchange
- **Expiry/TTL:** Enforced by wallet and partner API; short-lived by default

Privacy guarantee: Disclosures reveal facts (e.g., "origin KYC-verified," "balance ≥ X")—not raw transactions or counterparties.

3.4 Developer SDKs: dApp Adapters, Wallet API Hooks

Client APIs (TypeScript/Rust)

```
// Transparent operations
sendTransparentTx(txSpec) → standard Solana send

// Shielded operations
sendShieldedTx({
  outputs[],
  memo?,
  feeToken: NOC
}) → builds proof client-side; submits to verifier

// Shielded swap operations
executeShieldedSwap({
  inputToken: string,
  outputToken: string,
  inputAmount: string,
  minOutputAmount: string,
  slippageTolerance: number,
  feeToken: 'NOC'
}) → builds private swap proof; submits to shielded DEX aggregator

getShieldedSwapQuote({
  inputToken: string,
  outputToken: string,
  amount: string
}) → returns estimated output and fee in NOC without revealing identity

// Cross-mode transfers
crossModeTransfer({
  direction,
  amount
}) → deposit/withdraw helpers with unlinkability defaults

// Disclosure APIs
generateViewKey({scope, ttl})
revokeViewKey(viewKeyId)
issueAuditToken({auditor, scope, ttl, assertions[]})
```

Adapter Pattern

Drop-in wrappers for common flows (DEX swap, transfer, repay, NFT settle) with "private preferred" routing if supported.

Errors & Telemetry

Standardized codes `(ZK_PROOF_TIMEOUT, NULLIFIER_REPLAY, GEO_RESTRICTED, SWAP_SLIPPAGE_EXCEEDED, SWAP_INSUFFICIENT_LIQUIDITY)` and optional privacy-preserving telemetry for UX improvements (opt-in).

3.5 Compliance UX: Threshold Prompts, Geofencing, KYC Pointers

- **Threshold prompts:** Client checks soft/hard thresholds (e.g., value > regional cap). UI suggests creating an Audit Token or performing KYC when needed
- **Geo-fencing:** Wallet disables Shielded Mode in restricted jurisdictions based on IP + KYC country (defense-in-depth); transparent mode remains unaffected
- **KYC pointers:** For custodial ramps/exchanges, wallet can attach a KYC hash pointer within an Audit Token—verifiable by the partner without exposing identity data in-wallet
- **Travel-Rule adapters:** Optional integration paths to providers (e.g., Notabene/OpenVASP-style) triggered only for applicable flows

3.6 Security Model: Key Management, Local Secrets, Multi-Sig Options

Key Hierarchy

- **Public keys:** Standard Solana Ed25519 for transparent accounts
- **Shielded keys:** Derived spend/view keys for commitments/nullifiers; stored locally (Secure Enclave/Keychain/Keystore when available)
- **Disclosure branch:** Separate derivation path for View Keys/Audit Tokens to minimize blast radius

Local Security

- Encrypted at rest; biometric/PIN gating for spends; anti-phishing signer screens (human-readable tx intents)
- Optional passphrase shard (Shamir) for social recovery

Multi-sig / Enterprise

- Shielded multi-sig via MPC/threshold signing for spend authorization; view access scoped per role (finance, compliance, auditor)
- Policy engine: amount/day caps, mandatory dual-control on exits, auto-issuance of Audit Tokens for regulated flows

On-chain Safeguards

- Nullifier set to prevent double-spends; rate-limit guards; circuit versioning with upgrade gates
- Slashing hooks for misbehaving provers/relayers (liveness/validity), with rewards redistributed to honest participants

Operational Security

Formal audits (wallet + verifier + circuits), bug bounty, anomaly detection (e.g., unexpected nullifier patterns), and documented incident response with paused-mode fallback under governed controls.

4. Protocol Architecture - Shielded Privacy Layer on Solana

4.1 High-Level Design (Commitments, Nullifiers, Merkle State)

Model: Noctura maintains a shielded state on Solana defined by a commitment set, a nullifier set, and a Merkle root:

- **Commitment (C):** A binding of note contents (amount, recipient key, randomness) using a collision-resistant hash
- **Nullifier (N):** A one-time marker derived from a spent note's secret to prevent double-spends without revealing the note
- **Merkle Root (R):** The on-chain commitment tree root; membership proofs show a note exists in the tree

Transactions: A shielded transfer consumes one or more existing notes (proving membership and non-replay via nullifiers) and creates one or more new notes (emitting new commitments). Balances never appear in plaintext; only R, new Cs, and Ns are written on-chain.

Privacy guarantees: Sender, receiver, and amounts are hidden; linkability between inputs and outputs is computationally infeasible under the hash and ZK assumptions.

4.2 Off-Chain Proving, On-Chain Verification (State Anchoring)

Proving (off-chain)

Clients (or delegated provers) generate zk-proofs for:

- **Membership:** Each input note exists in the current (or recent) R
- **Non-replay:** Each input emits a unique N unlinked to the note
- **Balance conservation:** sum(inputs) = sum(outputs) + fees

Verification (on-chain)

A Solana verifier program checks proof validity, updates the Merkle tree with new commitments, and inserts nullifiers into the nullifier set atomically. Invalid proofs are rejected; duplicated Ns revert.

Anchoring & Finality

Every accepted batch updates R deterministically; Solana finality implies final shielded state after confirmation. Off-chain UIs track the canonical R and recent roll-forward path.

4.3 Proof System Strategy (SNARK-first; STARK optional roadmap)

- **SNARK-first:** Initial circuits use succinct zk-SNARKs (Groth16/PLONK family) to minimize verification cost and calldata size—well suited to frequent payments and swaps
- **Aggregation:** Support for proof aggregation (or succinct batch verification) reduces on-chain verifier cost per transfer as activity grows
- **STARK as option:** zk-STARKs (transparent setup, PQ posture) are an optional module for large audits or compliance proofs where bigger proof size is acceptable. They are not required for private payments and are avoided until a clear benefit outweighs complexity

4.4 Performance Constraints & Realistic Throughput Targets

Reality over slogans: zk-proving is compute-heavy; on-chain verification and account updates consume Solana compute units. "65,000 TPS" for fully shielded transfers is not realistic.

Launch Targets

- **Single-tx verification:** Tens of ms on-chain; practical ceiling limited by CU budget
- **Throughput (payments):** ~100–300 shielded TPS with baseline batching/aggregation under typical mainnet limits
- **Throughput (swaps):** Multi-token swaps require additional constraints for cross-asset balance proofs, increasing verification to ~250-300k CU (still economical at flat fee structure)
- **Latency:** ~1–3s median confirmation (proof ready) depending on client hardware/prover lanes

Scale Levers

- **Batching & aggregation:** Verify multiple transfers per proof to amortize verifier cost
- **Recursive proofs:** Aggregate many sub-proofs into one outer proof
- **Parallel provers & GPU lanes:** Reduce client/prover wait times; marketplace for throughput
- **Circuit hygiene:** Pedersen/Poseidon-style hashes; fixed-base MSM optimization; shallow trees for hot paths

Degradation Plan

Under load, the wallet dynamically shifts to larger batches (higher latency, higher TPS) with user-visible estimates.

Transaction fee quote exposed in wallet: $0.0005-$0.008 USD (converted to NOC) includes proof verification (~150-300k CU), prover/relayer compensation, and optional burn.

4.5 State Commitments & Data Availability (Anchors, Epochs, Finality)

- **Anchors & epochs:** Each verified batch references an anchor root (recent R) and produces a new root; epoch checkpoints (e.g., every N slots) are recorded for fast syncing and compact membership proofs
- **Tree management:** An append-only Merkle tree (sparse or incremental) is maintained on-chain; witnesses are updated off-chain. Clients can follow proof-carrying updates (PCU) to refresh paths without full re-sync
- **Data availability:** On-chain events emit new commitments, nullifiers, and the updated root; indexers (wallets, explorers) mirror these for users. No private data is required for liveness
- **Reorg handling:** In rare reorgs, clients re-anchor to the latest finalized root and revalidate pending proofs; the verifier rejects stale-anchor submissions

4.6 Prover/Relayer Participation & Incentives (Staked Roles, Slashing Hooks)

Roles

- **Provers:** Generate zk-proofs on behalf of users/dApps; selectable via a fee market
- **Relayers (optional):** Submit transactions and pay fees upfront for users who want extra network privacy; reimbursed in-protocol

Staking & Fees

- Operators stake $NOC to register; misbehavior risks slashing
- Fee market: users pick lanes by price × latency; wallet defaults to reputable operators
- Rewards paid in $NOC (and/or a portion of shielded fees); optional fee-burn introduces mild deflation

Slashing Conditions

- Invalid proofs / malformed batches → verifier-detectable faults
- Relayer theft/fraud (e.g., non-delivery) → challenge windows with evidence; slashed stake redistributed to affected users and honest operators

Liveness & QoS

Operators publish SLOs (max queue time, proof latency); wallet health checks route away from degraded lanes; DAO parameters can set minimum stake, maximum share, and emergency throttles.

Result: The architecture prioritizes provable privacy, realistic performance, and operational safety: succinct SNARK payments today, optional STARKs later; off-chain proving with on-chain finality; clear incentives and guardrails for the operators who keep the shielded layer fast and reliable.

5. Security Model & Threat Analysis

5.1 Adversary Model (Network, Cryptographic, Economic)

Network Adversaries

- **DoS/Spam:** Saturate verifier CUs or prover queues to degrade UX

- **Traffic analysis:** Correlate timing, size, and network paths to infer parties
- **Relayer abuse:** Censor, reorder, or front-run user submissions (for public exits)

Cryptographic Adversaries

- **Soundness attacks:** Attempt to forge zk-proofs / bypass nullifier checks
- **Collision attacks:** Target hash primitives (commitments, Merkle tree)
- **Key compromise:** Exfiltrate wallet keys (public or shielded); side-channel on client devices

Economic Adversaries

- **Operator misbehavior:** Provers/relayers deliver low quality, stall, or cheat fees
- **MEV-style inference:** Correlate shielded exits with public market activity
- **Sybil concentration:** Many small operators controlled by one actor to bias routing

Assumptions

- Standard hardness for chosen zk-proof system (SNARK today; STARK optional)
- Collision resistance of chosen hash (Poseidon/Pedersen family)
- Solana consensus/finality and L1 availability
- Honest-majority not required for privacy/soundness (verification is on-chain)

5.2 Formal Verification & Circuit Audits (Plan & Scope)

Specification First

- **State machine spec:** Commitments, nullifiers, Merkle updates, anchor rules, and rejection conditions
- **Conservation invariant:** Σinputs = Σoutputs + fees (mod base units)
- **Replay invariant:** Nullifier uniqueness is required and sufficient to prevent double-spends

Circuit Reviews

- **Math review:** Constraint systems, range proofs (amount), membership proofs (Merkle path), spend authorization
- **Witness handling:** Zeroization, bounds checks, domain separators, transcript binding
- **Test vectors:** Pathological inputs (duplicate paths, stale roots, zero-amounts, max amounts)

Program Verification

- **On-chain verifier:** Formal properties (reject-on-fail, atomicity of R, C[], N[] updates)
- **Tree logic:** Append, root rotation, epoch checkpoints
- **Access control:** Parameter governance, pause gates, upgrade keys with timelock

Audit Cadence

- **Phase I (presale/token):** Presale & token mint programs
- **Phase II (testnet privacy):** Verifier, tree ops, nullifier set
- **Phase III (pre-mainnet):** Circuits + wallet cryptography + SDK signing flows
- **Continuous:** Re-audit after circuit or parameter changes

5.3 Replay/Linkability Risks & Mitigations

Replay & Double-spend

- Nullifier set is on-chain; duplicates revert atomically
- Anchor freshness: proofs must reference a recent root within an epoch window; stale-anchor proofs rejected
- Nonce discipline: unique randomness per note & proof transcript binding

Linkability Vectors

- **Timing correlation:** Deposits/exits aligned in time → **Mitigation:** Randomized submit windows, relayer batching, optional delays
- **Amount correlation:** Exact amounts across modes → **Mitigation:** Output aliasing (split/join), wallet rounding bands, change outputs
- **Network metadata:** IP/ASN persistence → **Mitigation:** Relayers as privacy egress, optional Tor/VPN guidance, transport padding
- **UI fingerprints:** Deterministic behavior → **Mitigation:** Randomized fees within band, jittered batching thresholds

Exit Hygiene

Prefer fresh public addresses for exits; avoid immediate reuse; optional delay before exit settlement.

5.4 Runtime Monitoring, Anomaly Detection, Incident Response

Monitoring

- **On-chain signals:** Nullifier collision rate, proof failure ratio, root-advance cadence, verifier CU burn, queue depth
- **Operator SLOs:** Prover/relayer latency distributions, timeouts, error codes
- **Pattern detection:** Bursts near thresholds, unusual tree growth, repeated anchor staleness

Alerts & Thresholds

Automated alerts on: excessive proof rejects, root stagnation, spike in duplicate nullifiers, CU saturation, operator liveness failures.

Incident Response

- **Playbooks:** Categorization (P0–P3), communication templates, rollback/patch guidance
- **Controls:**
  - Rate limits (per account/per epoch) to throttle abuse
  - Emergency pause on shielded submit while preserving read access
  - Circuit/param rollback behind timelocked governance; hotfix key used only under DAO-defined conditions
- **Forensics:** Retain minimal, privacy-preserving logs (hashes, event IDs) for post-mortems; never store plaintext notes

5.5 Bug Bounties & Responsible Disclosure

Program Scope

- Wallet cryptography (key derivation, signing prompts, disclosure branch)
- Verifier programs (proof checks, tree ops, nullifier uniqueness)
- Circuits (soundness, constraint completeness, witness mishandling)
- SDK (transaction intent encoding, replay protection)
- Website/presale contracts (authZ/authN, funds safety)

Rewards & Process

- Tiered payouts by severity (RCE/Key exfiltration > forging spend > DoS > info leak)
- Safe harbor: good-faith research protected; coordinated disclosure timelines
- Hall of Fame: optional public credit, with consent
- Re-audit trigger: critical findings mandate third-party re-audit and changelog publication

Submission Hygiene

- Minimal PoCs preferred; no user data collection; reproduce with test vectors
- Provide environment details (SDK version, device/OS, RPC cluster)
- Use dedicated security email + PGP; response SLA published

Goal: Create a continuous security feedback loop where independent researchers, auditors, and the community can verify, and improve, the safety of Noctura's shielded layer without compromising user privacy.

6. Compliance & Selective Disclosure

6.1 Opt-In Privacy Model (Transparent by Default)

- **Default posture:** Wallet initializes in Transparent Mode (standard Solana account). Shielded Mode is explicitly enabled by the user with an in-wallet consent screen outlining jurisdictional limits and disclosure options
- **Granularity:** Privacy can be applied per-account or per-transaction. The wallet surfaces when a flow may require selective disclosure (e.g., fiat off-ramp, large exits, or private swaps above partner thresholds)
- **Data minimization:** The protocol never stores PII. Compliance is achieved via cryptographic attestations and user-controlled disclosures, not persistent identity records

6.2 View Keys & Audit Tokens (Interfaces, Scoping, Revocation)

Purpose: Allow users to prove facts (funds, origin, solvency) without exposing their full transaction graph.

A) View Keys (read-only, scoped)

- **Scope types:** `tx_id`, `time_window`, `balance_threshold`, `proof_of_funds`
- **Derivation:** From a disclosure branch key; cannot spend or reveal secrets
- **Revocation:** Local revocation + optional on-chain revocation hash partners must honor

Interface (wallet SDK):

```
type ViewScope =
  | { kind: "tx_id"; ids: string[] }
  | { kind: "time_window"; from: number; to: number }
  | { kind: "balance_threshold"; token: string; gte: string }
  | { kind: "proof_of_funds"; token: string; amount: string };

generateViewKey(scope: ViewScope, ttlSec: number):
  Promise<{ viewKeyId: string; key: string; expiry: number }>;

revokeViewKey(viewKeyId: string):
  Promise<{ revoked: boolean }>;
```

B) Audit Tokens (consent-bound assertions)

- **Payload (example):** `{ auditor_pubkey, scope, ttl, zk_assertions[], nonce, sig_user }`
- **Use cases:** KYC pointer proof, origin proof for exchange withdrawal, Travel-Rule partner checks
- **Expiry:** Short-lived by default; partners must refresh with user consent

Interface (wallet SDK):

```
type AuditAssertion =
  | { kind: "kyc_pointer"; hash: string }
  | { kind: "origin_clean"; policy: string }
  | { kind: "proof_of_funds"; token: string; gte: string };

issueAuditToken(
  auditorPk: string,
  scope: ViewScope,
  assertions: AuditAssertion[],
  ttlSec: number
): Promise<{ auditToken: string; expiry: number }>;
```

Partner verification (server API sketch):

```
POST /audit/verify
Body: { auditor_pk, audit_token, request: { tx_ids?, amount? } }
→ 200 { valid: true, zk_assertions: [...], scope: {...}, expires_at }
```

6.3 KYC / Travel-Rule Integrations (Partner Adapters)

- **KYC pointers (hash, not PII):** When a regulated partner requires identity, the wallet can attach a KYC pointer hash inside an Audit Token; the partner resolves it with their own records
- **Travel-Rule adapters:** Optional connectors (e.g., Notabene/OpenVASP-style) can be invoked only for covered transactions and only with user consent
- **Policy control:** Adapters are off by default; enabling requires an explicit toggle and a per-transfer prompt showing what will be disclosed

Adapter concept:

```
enableTravelRuleAdapter(
  providerId: "notabene" | "openvasp",
  configRef: string
): Promise<void>;

requestTravelRuleDisclosure(params):
  Promise<{ required: boolean; fields: string[] }>;
```

6.4 Geo-Fencing & Restricted Jurisdictions

- **Blocked regions list:** Maintained and updated in-app; includes hard bans and conditional allowances per legal counsel
- **Enforcement points:**
  1. App level: Shielded Mode toggle disabled; transparent mode unaffected
  2. Server adapters: Prover/relayer lanes can deny requests from restricted locales
  3. On-chain checks: Where applicable, program parameters may restrict specific flows
- **Signals used:** IP geolocation, user-declared country, and (if present) KYC country
- **Fail-safe:** If ambiguous, default to deny shielded / allow transparent

6.5 Compliance Transparency Page & Reporting

Public transparency page:

- Presale contract address: `6nTTJwtDuxjv8C1JMsajYQapmPAGrC3QF1w5nu9LXJvt`
- Current restricted jurisdictions and rule rationale
- KYC providers, audit firms, and links to audit reports (presale, token, verifier)
- DAO-ratified parameters: thresholds, fee-burn %, staking minimums

Operational metrics (privacy-preserving):

- Root update cadence, proof accept rate, duplicate nullifier rejects, verifier CU usage
- Adapter invocation counts (no PII), audit-token issuance totals by scope

Change control: Changelogs for circuit versions, parameter updates, and adapter configurations with effective dates.

Contacts: Security PGP, responsible disclosure policy, and legal inquiries channel.

Outcome: Compliance is delivered via user-consented, scoped proofs, not blanket deanonymization, backed by transparent policies, auditable artifacts, and opt-in integrations that keep Noctura usable in regulated markets.

7. Tokenomics

7.1 Supply & Allocation

Total Supply: 256,000,000 $NOC (fixed)

Distribution Model

Category	Percentage	Tokens	Unlock Details
Community	40%	102,400,000	Released through presale
Staking	20%	51,200,000	Reward pool for stakers
Liquidity	14%	35,840,000	DEX & CEX liquidity
Marketing	6%	15,360,000	Growth, outreach, strategic partners
Community Rewards	5%	12,800,000	Airdrops, contests, engagement
Team	8%	20,480,000	1.5 year lockup after TGE
Reserve	7%	17,920,000	Emergency & protocol upgrades
Total	100%	256,000,000	

Notes:

- Team tokens are locked for 18 months after launch, ensuring alignment with project success
- Reserve is for critical upgrades or unforeseen events and remains untouched unless approved by governance

7.1.1 Unlock & Vesting Schedule

- **Presale (40% / 102,400,000):** Locked until TGE, then claimable. Buyers may optionally lock into post-TGE staking tiers (365/182/90 days)
- **Team (8% / 20,480,000):** 18-month lock after TGE, followed by linear vesting over [specify period, e.g., 12-24 months]
- **Staking Rewards (20% / 51,200,000):** Released through emissions pool according to epoch-based distribution (see §7.4 & §9.3)
- **Liquidity (14% / 35,840,000):** [Specify unlock - e.g., "Released at TGE for DEX/CEX liquidity provisioning" or staged release]

- **Marketing, Community Rewards, Reserve:** Linear or cliff vesting based on allocation category with automatic release according to predetermined schedule

Note on Vesting Implementation: This version of the whitepaper does not publish an exact %-per-month (or full category-by-category) unlock schedule. Vesting follows predetermined schedules with no early withdrawal mechanisms to prevent manipulation. Detailed unlock schedules will be published on the Transparency Page prior to TGE with on-chain verification.

7.2 $NOC Utility - The Technical Foundation

Why $NOC is Technically Required (Not Just a Fee Token)

$NOC is not an arbitrary choice—it is **structurally required** for Noctura's privacy architecture to function. Here's why:

1. Shielded Transaction Fuel (Technical Necessity)

Private transactions cannot use variable external tokens for fees because:

- **Fee visibility problem:** If fees were paid in the same token being transferred (e.g., paying SOL fees to send private SOL), the fee amount would leak information about the transaction size
- **Cross-token complexity:** Supporting multiple fee tokens would require separate shielded pools per token, fragmenting liquidity and weakening anonymity sets
- **Unified privacy layer:** A single fee token (NOC) creates a uniform cost structure across all private operations, preventing fee-based correlation attacks

Technical implementation:

- All shielded operations (transfers, swaps, withdrawals) consume NOC for verification compute costs
- Fee amounts are concealed within zk-proofs (fee_commitment field)
- Verifier program validates fee payment before updating shielded state

2. Shielded Swap Mechanism (Core Utility)

When executing private swaps, $NOC serves multiple technical roles:

a) Privacy preservation: • Swap fees paid in NOC separate the fee layer from the trading layer
• Prevents linking swap amounts to fee amounts (which would leak trade size)
• Maintains unlinkability between input/output tokens

b) Liquidity routing: • NOC fees incentivize liquidity providers in shielded pools
• Aggregators earn NOC for routing optimal private swap paths
• Creates sustainable market for private liquidity without exposing volumes

c) Atomic swap verification: • zk-proofs verify swap balance conservation:
`input_token_amount = output_token_amount (at oracle price) + fee_NOC`
• On-chain verifier checks NOC fee commitment before accepting swap proof
• Prevents value leakage while maintaining privacy

3. Prover/Relayer Incentive Layer

Private operations require computational resources:

- **Proof generation:** Off-chain provers generate zk-proofs (CPU/GPU intensive)
- **Relay services:** Relayers submit transactions to break IP/timing correlation
- **NOC compensation:** Provers/relayers stake NOC and earn fees for honest service
- **Quality assurance:** Slashing mechanisms (in NOC) ensure reliable performance

Without NOC incentives, the prover network would not have economic alignment to maintain the privacy infrastructure.

4. Governance & Parameter Control

Network security requires economic skin-in-the-game:

- Staked NOC = voting power over protocol parameters (fee rates, batch sizes, circuit upgrades)
- Prevents hostile takeovers: Attackers must acquire and lock significant NOC to influence critical privacy parameters
- Aligned incentives: Voters have economic interest in maintaining privacy guarantees and network reliability

5. Fee Structure & Economic Flow

Flat fee model ($0.0005-$0.008 converted to NOC):

- Prevents transaction size correlation (all private ops cost the same)
- Low barrier to entry for privacy adoption
- Fee distribution:
    - 40-60%: Prover/relayer operators (computational costs)
    - 20-30%: Stakers (network security)
    - 10-20%: Protocol treasury (development/audits)
    - 0.25-1%: Optional burn (DAO-controlled deflationary mechanism)

Technical Example: Private Swap Flow

1. User initiates swap: 100 private USDC → SOL
2. Wallet generates swap proof showing:
    - Input: commitment to 100 USDC note
    - Output: commitment to X SOL note (amount hidden)
    - Fee: commitment to 0.008 NOC (fee hidden)
    - Balance conservation: verified in-circuit
3. Proof submitted to verifier via relayer (paid in NOC)
4. On-chain verification (~250-300k compute units)
5. State updated: input nullified, output committed, fee routed
6. User receives private SOL; no public trace of swap details

Why Not SOL or USDC for Fees?

Using external tokens would break privacy:

- **SOL fees** → Every transaction leaks SOL balance changes
- **USDC fees** → Trading stablecoins privately becomes impossible (circular dependency)
- **Multi-token fees** → Fragments liquidity, enables correlation attacks via fee token choice

$NOC solves this by being:

- **Privacy-native** (only used in private context)
- **Uniform** (same cost structure for all operations)
- **Aligned** (value accrues with privacy network usage)

Summary: NOC is Infrastructure, Not a Tax

$NOC is not a "fee for the sake of fees"—it's the cryptoeconomic primitive that makes private swaps, transfers, and compliance possible on Solana while maintaining strong unlinkability and sustainable network operations.

7.3 Transaction Fees

Transparent Mode: ~$0.0005

- Standard Solana-level fees
- Full DeFi/NFT compatibility
- Paid in SOL (no NOC required)

Shielded Mode: $0.0005 - $0.008 (converted to NOC at current rate)

- **Flat-rate privacy:** Same cost regardless of amount transferred/swapped
- **Covers computational costs:**
    - zk-proof verification (~150-300k compute units)
    - Merkle tree updates
    - Nullifier set writes
- Includes prover/relayer compensation
- Optional fee-burn (0.25-1% DAO-controlled)

Fee Breakdown by Operation

Private Transfers: $0.0005 in NOC

- Single-output shielded send
- Simple commitment/nullifier update
- ~150k CU verification cost

Private Swaps: $0.006 - $0.008 in NOC

- Multi-token zk-proof (input + output tokens)
- Liquidity routing fee
- Aggregator compensation
- ~250-300k CU verification cost

Cross-Mode Operations: $0.001 - $0.003 in NOC

- Deposit (public → shielded): $0.001
- Withdrawal (shielded → public): $0.002
- Includes unlinkability guarantee computation

Why Flat Fees?

Privacy preservation through uniform costs:

- Amount-based fees would leak transaction size information
- Flat structure prevents correlation between fee and value transferred
- Encourages privacy adoption (no penalty for large transactions)
- Predictable UX (users know exact cost before signing)

Fee Payment Mechanism

1. Wallet estimates NOC requirement at current USD/NOC rate
2. User confirms operation with fee shown in both NOC and USD
3. Fee concealed in zk-proof (fee_commitment)
4. Verifier deducts NOC from shielded balance
5. Distributed to operators/stakers per emissions policy

Note: Presale buyers receive zero-fee transactions (both modes) for 18 months post-launch.

7.4 Emissions Policy & Treasury Controls (DAO-governed)

A. Emissions (Staking & Operators)

- **Pool cap:** 51,200,000 $NOC total (20% of supply)
- **Curve:** Front-weighted but decaying emissions with epoch updates; parameters on-chain (epoch length, base rate, decay)
- **Operator share:** Portion of emissions/fees earmarked for prover/relayer operators proportional to delivered QoS (latency, reliability)
- **Auto-adjust:** If total staked rises, APR falls proportionally; if it falls, APR rises—keeping issuance predictable

B. Staking Tiers (defined in Staking section)

Post-TGE APR options are enforced via lock contracts (e.g., 128%/365d, 68%/182d, 34%/90d), with DAO authority to tune base rates and lock multipliers within published bounds.

C. Treasury Policy

- **Multisig + Timelock:** Treasury movements require multi-party approval and time-delayed execution; all txs labeled and disclosed
- **Budget rails:** % caps for liquidity, audits, integrations, and grants; any override requires DAO vote
- **Stability tools:** Treasury may supply/withdraw liquidity to meet depth targets (slippage caps), not for price support
- **Reporting:** Quarterly on-chain reports: remaining runway, emissions paid, fee-burn totals, grants issued

D. Parameter Governance

DAO motions (quorum/threshold defined in Governance):

- Emission decay/epoch length; operator reward split
- Fee-burn % and fee bands
- Minimum operator stake & slashing schedule
- Circuit version activation (post-audit)

Safety rails: Emergency pause and parameter ceilings/floors are hard-coded and can only be changed by supermajority with a timelock.

Outcome: $NOC economics are usage-driven (fees + operators), transparent (on-chain parameters, reports), and governable (DAO), balancing incentives for users, operators, and long-term protocol resilience.

8. Presale Structure

The Noctura presale is structured to reward early adopters and foster a wide community.

A total of **102,400,000 $NOC** (40% of total supply) will be distributed across 10 stages, with a gradually increasing price per stage to incentivize early participation.

Presale Highlights

- **Fair Launch:** No VCs, no insiders, no early privileges — just equal access for everyone through our transparent presale
- **Multi-stage Incentives:** Early buyers receive the lowest price, creating organic demand and buzz

Vesting Implementation

- Presale tokens locked until TGE
- Linear or cliff vesting based on allocation category
- Automatic release according to predetermined schedule
- No early withdrawal mechanisms (prevents manipulation)

Presale Pricing Structure

Stage	Price ($)	Stage Increase (%)	Cumulative Gain (%)	Tokens per Stage
1	$0.1501	0.00%	0.00%	10,240,000
2	$0.1723	14.79%	14.79%	10,240,000
3	$0.1945	12.88%	29.58%	10,240,000
4	$0.2167	11.41%	44.37%	10,240,000
5	$0.2389	10.24%	59.16%	10,240,000
6	$0.2611	9.29%	73.95%	10,240,000
7	$0.2833	8.50%	88.74%	10,240,000
8	$0.3055	7.84%	103.53%	10,240,000
9	$0.3277	7.27%	118.32%	10,240,000
10	$0.3499	6.77%	133.11%	10,240,000

Accepted Payment Options

The presale supports multiple blockchain networks and payment methods to maximize accessibility:

- **Solana Network:** SOL, USDT (Solana), USDC (Solana)
- **Ethereum Network:** ETH, USDT (Ethereum), USDC (Ethereum)
- **BNB Chain:** BNB, USDT (Binance Chain), USDC (Binance Chain)
- **Fiat Payments:** Secure card payment option available for non-crypto participants

Purchase Limits

- **Minimum Buy:** $25 USD
- **Maximum Buy:** $25,600 USD

This on-chain, 10-stage structure provides transparent pricing, exact arithmetic to the cent, and clear compliance hooks—ready for investors, auditors, and exchange review.

8.1 Referral Program (Community Growth Incentive)

To further encourage organic adoption and community participation, Noctura introduces a **Referral Program** during the presale phase.

Each registered participant receives a unique referral link that can be shared with others. When a new buyer participates in the presale through a referral link, the referrer automatically earns a **10% bonus in $NOC tokens**, calculated based on the total value of the referred purchase.

Program Highlights

- **Reward:** 10% of the referred purchase amount, paid in $NOC tokens
- **Distribution:** Bonuses are automatically credited to the referrer's wallet after the referred transaction is confirmed
- **Eligibility:** Available to all verified presale participants

This referral program aligns with Noctura's vision of decentralized community growth and equitable participation while maintaining full transparency and auditability.

9. Staking & Rewards

9.1 Pools, Eligibility, and Slashing (Provers/Relayers)

Pools

- **Stake Pool — Users:** Non-custodial pool for $NOC holders. Locks enforce APR tier (see §9.2)
- **Operator Pool — Provers/Relayers:** Operators must stake $NOC to register lanes and earn a share of protocol fees; subject to slashing

Eligibility

- Wallet signature + lock commitment on-chain; address-bound
- Presale stakers auto-migrate to post-TGE tiers at claim (no action required)
- Operators publish service endpoints and SLOs (latency, availability) signed with their staked identity

Slashing (operators)

- **Validity faults:** Malformed/invalid proof batches submitted → proportional slash
- **Liveness faults:** Repeated SLO violations (timeouts, abandonment) → escalating penalties
- **Fraud/replay:** Double-spend relay attempts, user funds theft → maximum slash + registry ban
- Slashed funds are redistributed to honest stakers and/or burned per DAO policy

9.2 APR Options

APR tiers are fixed-rate targets for the lock duration, funded from emissions and fees. The DAO may adjust tier rates/limits within published bounds

Tier	Lock	Target APR (nominal)	Notes
A	365 days	128%	Highest reward; longest lock; priority in fee rebates
B	182 days	68%	Medium lock/return
C	90 days	34%	Shortest lock; flexible entry/exit cadence

Notes:

- APRs are nominal targets funded from the emissions pool + protocol fees; actual realized APY depends on compounding choice and epoch timing
- Operator fees and fee-burn (if enabled) can shift effective yield; all parameters are on-chain and transparent

9.3 Reward Funding, Emission Curves, and DAO Parameters

Funding Sources

- **Emissions pool:** 51,200,000 $NOC (20% of supply) allocated to staking & operators
- **Protocol fees:** Portion of shielded transaction fees routed to stakers and operator lanes
- **Optional fee-burn:** 0.25–1.00% of fees burned (DAO-set) for deflationary pressure

Emission Mechanics

- **Epoch-based accounting** (e.g., daily): rewards minted/credited per epoch
- **Auto-balancing:** If total staked rises, per-unit rewards fall (and vice-versa) to keep issuance within budget
- **Curve stewardship:** DAO can tune epoch length, base rate, and operator split within guardrails (caps, floors, timelock)

Operator Share

A fixed slice of epoch rewards (or fees) pays registered provers/relayers based on delivered throughput/reliability (oracle-reported SLOs).

Transparency

- **Live dashboard:** Emissions spent, fee revenue, burn totals, and pool APRs; historical CSVs/IPFS snapshots published quarterly

9.4 Unstaking & Cool-Down, Auto-Compound, Fee-Burn Tie-ins

Unstaking

- **Lock-enforced:** Funds are unlockable only after the chosen lock term (365/182/90 days)
- **Cool-down:** A 48-hour cool-down applies before withdrawal to deter flash-stake gaming
- **Early exit:** Not permitted (principal protected); DAO may define emergency early-exit with penalty in extraordinary circumstances

Reward Handling

- **Auto-compound (opt-in):** Reinvest epoch rewards into the same lock band to increase effective APY

- **Manual claim:** Claimable at any time without resetting lock; claimed rewards follow wallet's current mode (transparent by default)

Fee-burn Integration

If fee-burn is enabled, burn events occur before reward splits, slightly reducing inflation and supporting long-term value accrual.

Security & Operations

All staking contracts are audited, parameter changes are timelocked, and emergency pause affects new stakes only; existing stakes continue accruing per schedule.

Outcome: Staking aligns users, operators, and protocol health: clear lock-based APR choices for holders, service-quality incentives for operators, and DAO-governed emissions that stay within budget and adapt as real fee revenue grows.

10. Technical Feasibility Notes - Corrections & Rationale

10.1 Throughput Reality: Why "65,000 TPS shielded" is unrealistic

- **ZK cost isn't free:** Even with off-chain proving, on-chain verification consumes compute units (CUs), account writes (Merkle root, nullifiers), and transaction bandwidth
- **Account update bound:** Each shielded spend touches: verifier program, nullifier set, Merkle tree accounts, and fee accounts. These state writes, not just proof checks, cap throughput
- **Practical ceiling (launch):** With per-tx verification + tree updates, we target ~100–300 shielded TPS under typical mainnet limits. Higher TPS requires batching/aggregation (below)
- **Claim withdrawn:** We do not claim "65,000 TPS shielded." That figure ignores on-chain state and verification costs

10.2 Proving & Verification Bottlenecks (Cost Model; Batch/Aggregate/Recursive Proofs)

Cost Model (simplified)

- **Proving time (client/prover):** `T_prove ≈ α·C_circuit + β·log2(MerkleDepth)`
- **Verify time (on-chain):** `T_verify ≈ γ·Constraints + δ·verify_key_load + ε·proof_elements`
- **State cost:** `W_state ≈ writes(nullifiers + commitments + root)`

Performance Strategies

- **Single-tx verification** yields predictable latency but lower TPS due to fixed per-tx costs
- **Batch verification:** Combine k transfers into one proof → amortizes T_verify and W_state per transfer; increases latency per user slightly
- **Aggregation/recursive proofs:** Aggregate many sub-proofs into one outer proof; best TPS per CU, at the expense of prover complexity and longer batch windows
- **Swap verification cost:** Multi-token swaps require additional constraints for cross-asset balance proofs, increasing verification to ~250-300k CU (still economical at $0.006-$0.008 flat fee)

Conclusion: We start with single-tx + small batches; graduate to aggregation as demand grows.

10.3 Proof System Simplification: SNARK-first; STARKs optional

- **SNARK-first (Groth16/PLONK family):** Small proofs, cheap verification, well-understood tooling. Ideal for frequent private payments and swaps
- **Why not dual-stack at launch?** Maintaining both SNARK + STARK circuits doubles engineering and audit scope with limited initial benefit
- **STARKs (optional module):** Considered for large audit assertions or post-quantum posture where proof size is acceptable and transparency is valued. Introduced only when there is a clear, measured benefit and dedicated budget for audits

10.4 On Unlinkability: ZK already provides it; "Advanced Mixing" is redundant

- Robust shielded transfers (commitments/nullifiers + balance conservation + membership proofs) already break input→output linkability
- Mixing adds little beyond timing/fee smoothing and larger anonymity sets; it does not replace sound ZK design
- **Policy:** Noctura does not depend on mixer semantics for privacy. If ever used, mixing-like batching is opt-in UX (fee/timing), not the anonymity primitive

10.5 Performance Optimizations Roadmap (Batching, Aggregation, GPU Provers, Caching)

Phase A — Launch

- Circuit hygiene (Poseidon/Pedersen), fixed-base MSM optimization
- Merkle depth tuned for proof speed vs. anonymity set
- Modest k-batch verification (e.g., 4–8 transfers/proof)

Phase B — Scale

- Proof aggregation/recursive SNARKs to amortize on-chain verification
- GPU/ASIC prover lanes; prover marketplace with price×latency routing
- Witness caching & incremental path updates (PCU) for faster re-proofs

Phase C — Maturity

- Adaptive batch sizing (load-aware), epoch checkpoints for fast sync
- Optional STARK module for large, infrequent attestations
- DAO-tuned parameters: batch windows, operator minimum stake, fee bands

Bottom line: We ship SNARK-first with conservative, real-world throughput; scale via batching and aggregation; and only add complexity (e.g., STARKs, mixer-style batching) when the measured benefit justifies the audit and maintenance cost.

11. Developer References — API Schemas, zk Proof Formats

11.1 Wallet API (Transparent vs Shielded Ops; Params & Error Codes)

Ops (TypeScript)

```
// Transparent (public) operations
sendTransparentTx(tx: {
  to: string;
  mint: string;
  amount: string;
  memo?: string
}): Promise<TxId>;

getTransparentBalance(owner: string, mint: string): Promise<string>;

// Shielded (private) operations
sendShieldedTx(tx: {
  outputs: Array<{
    pk_shielded: string;
    mint: string;
    amount: string;
    memo?: string
  }>;
  feeMint?: "NOC";
  anchor?: string;
  prover?: string;
  maxLatencySec?: number;
}): Promise<{ txId: TxId; newRoot: string }>;

// Shielded swap operations
executeShieldedSwap(params: {
  inputToken: string;
  outputToken: string;
  inputAmount: string;
  minOutputAmount: string;
  maxPriceImpact: number;
  deadline?: number;
  routePreference?: 'best_price' | 'least_slippage' | 'fastest';
  feeToken: 'NOC';
```

```
}): Promise<{
  txId: string;
  outputCommitment: string;
  actualOutputAmount: string;
  feesPaid: string;
  priceImpact: number;
}>;

getShieldedSwapQuote(params: {
  inputToken: string;
  outputToken: string;
  inputAmount: string;
  slippageTolerance?: number;
}): Promise<{
  estimatedOutput: string;
  priceImpact: number;
  feeInNOC: string;
  feeInUSD: string;
  route: Array<{pool: string; share: number}>;
  expiresAt: number;
}>;

getShieldedSwapHistory(params: {
  viewKey: string;
  fromTimestamp?: number;
  toTimestamp?: number;
}): Promise<Array<{
  txId: string;
  timestamp: number;
  inputToken: string;
  outputToken: string;
  inputAmount: string;
  outputAmount: string;
  feesPaid: string;
}>>;

crossModeTransfer(req: {
  direction: "public_to_shielded" | "shielded_to_public";
  mint: string;
  amount: string;
  recipient?: string;
}): Promise<TxId>;

getShieldedBalance(ownerViewKey: string, mint: string):
Promise<string>;

getRecentAnchor(): Promise<string>;

// Selective disclosure
generateViewKey(scope: ViewScope, ttlSec: number):
  Promise<{ viewKeyId: string; key: string; expiry: number }>;
```

Common Error Codes

```
ZK_PROOF_TIMEOUT, ZK_PROOF_INVALID, ANCHOR_STALE, NULLIFIER_REPLAY,
GEO_RESTRICTED, KYC_REQUIRED, SLA_UNAVAILABLE, FEE_QUOTE_EXPIRED,
INSUFFICIENT_FUNDS, MINT_UNSUPPORTED, RATE_LIMITED, INTERNAL_ERROR,
SWAP_INSUFFICIENT_LIQUIDITY, SWAP_SLIPPAGE_EXCEEDED,
SWAP_DEADLINE_PASSED,
SWAP_ROUTE_UNAVAILABLE, SWAP_QUOTE_EXPIRED, SWAP_TOKENS_IDENTICAL
```

11.2 Transaction/State APIs (Commitments, Nullifiers, Merkle Paths)

Indexer/Node (REST/JSON)

```
GET /v1/state/anchor
→ { root: "0x<32>", slot: 259921430 }

GET /v1/state/commitments?fromSlot=...&toSlot=...
→ { items: [{ C: "0x<32>", slot: ..., idx: ... }], nextPage: ... }

GET /v1/state/nullifiers?fromSlot=...&toSlot=...
→ { items: [{ N: "0x<32>", slot: ... }], nextPage: ... }

POST /v1/state/merkle-path
Body: { C: "0x<32>", root: "0x<32>" }
→ { path: ["0x<32>", ...], indices: [0|1,...], depth: 32 }
```

Event Topics (on-chain logs)

```
ShieldedCommit(C, newRoot, slot)
NullifierSet(N, slot)
RootRotated(oldRoot, newRoot, epochId)
```

11.3 zk Proof Object Formats (commitment, nullifier, merkle_root, proof_bytes)

Note & Commitment (conceptual)

```
note = (pk_recipient, mint, amount, r)
C = H(pk_recipient || mint || amount || r)  // Poseidon/Pedersen
N = PRF(sk_spend, note_id)  // unlinkable nullifier
```

Spend Proof (binary envelope → base64/hex in APIs)

```
type SpendProof = {
  // Public inputs
  merkle_root: string;
  nullifiers: string[];
  commitments: string[];
  fee_commitment: string;
  memo_commitment?: string;

  // Proof bytes
  proof: string;
  vk_id: string;
  anchor_slot: number;
};
```

Verification Constraints (summary)

- `merkle_path(note) → merkle_root`
- All nullifiers unique & not in set
- `Σinputs = Σoutputs + fee` (range-checked)
- Authorized spend under `sk_spend` (witness)

11.4 Verifier Program Interface (On-Chain) & Gas/Fee Estimates

Solana Instruction Layout (Anchor-like pseudo-IDL)

```
pub fn submit_shielded_batch(
  ctx: Context<SubmitBatch>,
  proof: Vec<u8>,
  vk_id: [u8; 32],
  merkle_root: [u8; 32],
  nullifiers: Vec<[u8; 32]>,
  commitments: Vec<[u8; 32]>,
  fee_commitment: [u8; 32],
  anchor_slot: u64,
) -> Result<()>;
```

Accounts

- **VerifierState** (holds current root, vk registry, epoch params)
- **NullifierSet** (probabilistic or bitmap shards; write-once)
- **CommitmentTree** (incremental tree nodes)
- **FeeVault**, **Treasury**, **OperatorRegistry**

Fee/Compute Guidance (targets, not promises)

- Single-spend tx verify + state writes: ~150–300k CU
- Small batch (4–8 spends): ~350–700k CU
- Tx fee quote exposed in wallet (includes L1 fees + prover/relayer fee)

11.5 Audit/Disclosure API (Scopes, ZK Assertions, Consent Tokens)

Scopes

```
type ViewScope =
  | { kind: "tx_id"; ids: string[] }
  | { kind: "time_window"; from: number; to: number }
  | { kind: "balance_threshold"; mint: string; gte: string }
  | { kind: "proof_of_funds"; mint: string; amount: string };
```

Audit Assertions

```
type AuditAssertion =
  | { kind: "kyc_pointer"; hash: string }
  | { kind: "origin_clean"; policy: string }
  | { kind: "proof_of_funds"; mint: string; gte: string };
```

Token Issuance & Verify

```
issueAuditToken(
  auditorPk: string,
  scope: ViewScope,
  assertions: AuditAssertion[],
  ttlSec: number
): Promise<{ token: string; expiry: number }>;

POST /v1/audit/verify
Body: { auditor_pk, token }
→ { valid: boolean, scope, assertions, expires_at }
```

Revocation

```
POST /v1/audit/revoke
Body: { token_id }
→ { revoked: true }
```

11.6 Example Flows (Transparent→Shielded, Private Payment, Scoped Audit, Private Swap)

A) Transparent → Shielded (Deposit)

1. `getRecentAnchor()`
2. `crossModeTransfer({ direction: "public_to_shielded", mint, amount })`
3. Wallet builds SpendProof (membership of deposit UTXO, fee concealment), submits `submit_shielded_batch`

B) Private Payment (Shielded → Shielded)

1. Sender builds `outputs { pk_shielded_recipient, mint, amount }` (+ change)
2. `sendShieldedTx({ outputs, feeMint: "NOC", anchor })`
3. Verifier checks proof, inserts nullifiers, appends new commitments, rotates root

C) Shielded → Transparent (Withdrawal)

1. `crossModeTransfer({ direction: "shielded_to_public", mint, amount, recipient })`
2. Proof shows membership & nullifier uniqueness; L1 program releases funds to recipient

D) Scoped Audit (Proof-of-Funds)

1. Exchange requests: "prove ≥ X without history"
2. User: `issueAuditToken({ auditorPk, scope: {kind:"proof_of_funds", mint, amount:X}, assertions: [...], ttlSec })`
3. Exchange: `POST /v1/audit/verify` → receives `valid`, `scope`, `assertions`; no raw tx graph exposed

E) Private Swap (Shielded → Shielded)

Scenario: User wants to swap 1000 private USDC for SOL without revealing trade details

Flow:

1. **Get swap quote:**

```
const quote = await getShieldedSwapQuote({
  inputToken: "USDC_mint",
  outputToken: "SOL_mint",
  inputAmount: "1000",
  slippageTolerance: 0.5
});
// Returns: ~47.2 SOL estimate, 0.008 NOC fee, 0.3% price impact
```

2. **Execute private swap:**

```
const result = await executeShieldedSwap({
  inputToken: "USDC_mint",
  outputToken: "SOL_mint",
  inputAmount: "1000",
  minOutputAmount: "47.0",
  feeToken: "NOC"
});
```

3. Behind the scenes:
    - Wallet generates multi-token swap proof:
        - Input: nullifier for 1000 USDC note
        - Output: commitment for ~47.2 SOL note
        - Fee: commitment for 0.008 NOC
        - Balance conservation verified in-circuit
    - Proof routes through aggregator to optimal private liquidity
    - Verifier checks proof, updates state atomically
    - User receives SOL in shielded balance

4. Privacy guarantees:
    - No on-chain trace of swap amounts
    - Input/output tokens unlinkable
    - Trading strategy concealed
    - Only user (with view key) can see swap details

Note: Swap fees are paid in NOC to prevent fee-amount correlation that would leak transaction size.

Versioning & Stability

- All objects carry `vk_id/schema_v`. Circuit or API changes bump versions; old paths deprecate with notice
- Open-source IDLs/SDKs are published with test vectors and conformance tests

12. Roadmap

Stage 1 - Foundation & On-Chain Presale

Deliverables

- On-chain presale live (10 stages), transparency page, team KYC published
- Presale & token programs audited; incident-response + timelock upgrade playbooks
- Verifier program skeleton + Merkle/Nullifier accounts on devnet (no funds risk)
- Wallet SDK preview (transparent ops + stubs for shielded ops)
- Selective Disclosure v0 (View Key/Audit Token schemas, mock verifier)
- Operator registry spec (staking & SLOs) and slashing policy draft

End-of-Stage Goal: Noctura Wallet Beta (Testnet)

Dual-mode UI (Transparent/Shielded toggle), deposit/withdraw flows wired to test contracts, basic proofs via hosted prover lane, scoped View Key issuance.

Ready-to-Ship Criteria

- Presale contract: audit signed; public contract address posted
- Wallet Beta passes smoke tests on testnet; public test build + docs

Stage 2 - Devnet Shielded Pools; Alpha Prover Lanes

Deliverables

- Shielded pools live on devnet with real circuits (SNARK-first)
- Small-batch verification (4–8 spends/proof); epoch checkpoints
- Prover/relayer alpha marketplace (price × latency), operator staking MVP
- Selective Disclosure v1 (scoped View Keys, Audit Tokens; verify endpoint)
- Compliance adapters (Travel-Rule/KYC pointer) behind opt-in toggles

Ready-to-Ship Criteria

- ≥3 external operators staked; latency/throughput SLO dashboards live
- End-to-end private payment demo (devnet) with scoped audit at partner sandbox

Stage 3 - Testnet Verifiers; Wallet RC (Dual-Mode)

Deliverables

- Verifier program on public testnet; batching + aggregation benchmarks
- Wallet Release Candidate with full dual-mode flows and cross-mode unlinkability
- **Shielded swap integration** (private token exchange beta)
- Proof aggregation roadmap; GPU prover lanes pilot; witness caching (PCU)
- Security pass: circuit review, verifier review, SDK signing flows audit

Ready-to-Ship Criteria

- Private payments at 100–300 shielded TPS (testnet), 1–3s median latency
- **Private swaps functional** with quote/execution/history APIs
- Bug bounty program launched; P0/P1 triage + disclosure SLAs published

Stage 4 - Mainnet Shielded Layer; Listings & Partners

Deliverables

- Mainnet launch of Shielded Privacy Layer (SNARK-first)
- **Shielded swaps live** with aggregated private liquidity
- Staking program live (post-TGE tiers: 128%/365d, 68%/182d, 34%/90d)
- Fee routing (stakers + operators), optional fee-burn parameter exposed to DAO
- Initial CEX/DEX liquidity (from 14% allocation), partner integrations (wallets/DEXs)

Ready-to-Ship Criteria

- Two independent audits signed (verifier + circuits)
- ≥2 exchange listings (one tier-1/2), ecosystem partner live (DEX or lender)
- **Private swap volume tracking** enabled with privacy-preserving metrics

Stage 5 - Enterprise Console; DAO Bootstrap; Grants

Deliverables

- Enterprise Console: policy engine (limits, dual-control), scoped view management
- DAO bootstrap: quorum/thresholds, parameter caps/floors, timelock governance
- Grants program (privacy dApps, explorers, analytics with privacy guarantees)
- Transparency: quarterly emissions/fees/burn reports; circuit versioning policy

Ready-to-Ship Criteria

- First DAO votes executed (operator min-stake, fee-burn %, batch windows)
- 3rd-party ecosystem apps using the Audit/Disclosure API in production
- **Private DeFi integrations** (lending, yield, swaps) live

Stage 6 - Scale-Out; Cross-Chain Privacy; Institutional Pilots

Deliverables

- Throughput scale: aggregation/recursive SNARKs enabled; adaptive batch sizing
- Cross-chain privacy R&D (bridged attestations; STARK module for large audits if justified)
- Institutional pilots (treasury/payroll, PoF attestations with custodians)
- Operator marketplace expansion, SLO enforcement & slashing in production

Ready-to-Ship Criteria

- Sustained mainnet throughput improvements; operator churn < X%/quarter
- Pilot case studies published; measurable fee revenue covering ≥Y% emissions

KPI Hints (internal, optional to publish)

- **Stage 1:** ≥10k testnet wallet installs; presale progress by stage
- **Stage 2–3:** ≥100–300 shielded TPS testnet; ≥3 external operators
- **Stage 4:** Mainnet MAU ≥25k; shielded tx share ≥20% of wallet tx; **swap volume ≥ $1M/week**
- **Stage 5–6:** DAO participation rate ≥10% of staked supply; fee revenue covers ≥30–50% of emissions

13. Investor Benefits & Presale Rationale

13.1 Why Invest in $NOC Now

Noctura delivers the **first shielded privacy layer on Solana** with a dual-mode wallet (Transparent ⇄ Shielded). It's engineered for usable privacy and selective disclosure, making it compatible with exchanges and regulated partners while preserving strong confidentiality on-chain.

Why $NOC at this stage:

Wallet-first adoption engine

Privacy is not a separate chain—it's a toggle in the wallet. That UX, plus SDK hooks for dApps, is the fastest path to real usage.

Private DeFi breakthrough

Shielded swaps enable confidential trading without revealing positions or strategies—solving front-running, MEV extraction, and competitive intelligence leaks that plague transparent DeFi.

Compliance-ready by design

View Keys and Audit Tokens let users prove facts (proof-of-funds, origin) without exposing their history—making Noctura far more exchange-friendly than legacy privacy tools.

SNARK-first, realistic performance

Off-chain proving with on-chain verification targets hundreds of shielded TPS at launch, scalable via batching/aggregation and GPU provers. No hype TPS; only what Solana and ZK can credibly support.

Direct economic flywheel

Every shielded transfer, private swap, and operator lane is powered by $NOC—structurally required for privacy-preserving fee handling and prover compensation, not an arbitrary fee token. All private operations (transfers, swaps, cross-mode) consume NOC, creating direct utility tied to network usage.

Ultra-low fees = mass adoption

At $0.0005-$0.008 per private operation (vs competitors at $0.05-$0.25+), Noctura makes privacy accessible for everyday transactions, not just high-value transfers. Flat-rate fees prevent transaction size correlation while maintaining economic sustainability.

Institutional fit

Dual-mode wallet + selective disclosure enables payroll, treasury, compliance attestations, and private DeFi—use cases that Monero/Zcash/mixers struggle to support in regulated venues.

On-chain presale, transparent math

10 fixed stages (4-dec prices) sum exactly to $25.6M hard cap; 40% supply allocated to presale. Parameters are public and auditable.

Presale Buyer Benefit: Zero $NOC transaction fees on both modes for 18 months post-wallet launch.

Why Noctura Succeeds Where Others Failed

Feature	Monero	Zcash	Tornado Cash	Noctura
Exchange Compatibility	❌ Delisted	⚠️ Limited	❌ Sanctioned	✅ Designed for listings
Compliance Tools	❌ None	⚠️ Minimal	❌ None	✅ Selective disclosure
User Experience	⚠️ Complex	⚠️ Complex	⚠️ Technical	✅ One-tap privacy
Scalability	⚠️ Moderate	⚠️ Low	⚠️ Limited	✅ Solana-native speed
Institutional Adoption	❌ Blocked	⚠️ Limited	❌ Impossible	✅ Enterprise-ready
Ecosystem Integration	❌ Standalone	❌ Standalone	⚠️ ETH only	✅ Full Solana ecosystem
Private Swaps	❌ No	❌ No	❌ No	✅ Built-in
Fee Structure	⚠️ Variable	⚠️ Variable	⚠️ Variable	✅ Ultra-low flat

Bottom line: Noctura is privacy that can survive in the real world—usable, verifiable, and partner-friendly, on Solana's performance rails.

14. FAQ

Q1: What is Noctura, exactly?

Noctura is the first shielded privacy layer on Solana with a dual-mode wallet. Users can transact in Transparent (public) or Shielded (private) mode, execute private swaps, and selectively disclose facts (e.g., proof-of-funds) without exposing their full history.

Q2: Is this a Layer-2/rollup?

No. It's a privacy overlay, not an L2 rollup. Proofs are generated off-chain and verified on-chain by Solana programs that update shielded state (commitments, nullifiers, Merkle root).

Q3: How is this different from Monero, Zcash, or Tornado Cash?

- **Monero:** Strong privacy, limited selective disclosure → exchange friction. Noctura provides view keys/audit tokens for scoped, consented proofs

- **Zcash:** Has view keys, but adoption hampered by UX fragmentation. Noctura's dual-mode wallet makes switching trivial
- **Tornado Cash:** Mixer semantics are non-compliant by default. Noctura's ZK-native unlinkability avoids mixer heuristics and supports compliance-ready attestations
- **All three:** No private swap functionality; Noctura enables confidential DeFi trading

Q4: What's the realistic private TPS?

Launch targets are ~100–300 shielded TPS, with 1–3s median confirmation, scaling via batching, aggregation, and GPU provers. We do not claim "65k TPS shielded."

Q5: Do I need KYC to use Noctura?

Transparent mode: typically no. Shielded mode or presale participation may require KYC/geo checks depending on your jurisdiction and thresholds. The protocol stores no PII—partners hold their own records; the wallet uses hash pointers inside audit tokens when needed.

Q6: How does selective disclosure work?

Users can issue View Keys (read-only, scoped by tx/time/balance) and Audit Tokens (consent-bound, expiring assertions like proof-of-funds or KYC pointer). Partners verify these without accessing the full transaction graph.

Q7: Is the presale on-chain? Where?

Yes. Presale contract (Solana):
`6nTTJwtDuxjv8C1JMsajYQapmPAGrC3QF1w5nu9LXJvt`

Q8: How is the presale priced?

10 stages, each selling 10,240,000 $NOC at 4-decimal fixed prices that sum to an exact $25,600,000 hard cap. Stage-to-stage increases are ~6–12%, with Stage-10 ≈ +133% vs Stage-1.

Q9: When and how do I claim tokens?

At TGE, via the Noctura dApp. You can optionally auto-stake into a lock tier at claim.

Q10: Does staking start during presale?

Yes—staking opens at presale start. Presale positions accrue eligibility and roll forward to post-TGE options; principal/rewards are claimable at/after TGE.

Q11: What are the staking options after TGE?

- 128% APR — 365-day lock
- 68% APR — 182-day lock
- 34% APR — 90-day lock

APRs are funded by emissions + protocol fees and governed by the DAO within published bounds.

Q12: Is there a presale boost (256% APR)?

If enabled, it's time-boxed to the presale accrual window only and stops at TGE; rewards are claimable after the post-launch unlock. Post-TGE, the fixed lock-tier APRs apply (see Q11).

Q13: What gives $NOC real utility?

$NOC powers shielded fees, private swap execution, prover/relayer incentives, staking/DAO governance, and premium wallet features (e.g., instant proving lanes). Optional fee-burn (DAO-set) can add mild deflation tied to usage.

Q14: How are tokens allocated?

Total supply 256,000,000 $NOC: 40% Presale, 20% Staking & Rewards, 14% Liquidity, 8% Core Team (18-month lock + vest), 7% Reserve/Upgrades, 6% Marketing/Partnerships, 5% Community Engagement.

Q15: Are contracts and circuits audited?

Yes. Audits cover presale/token programs, verifier/tree logic, wallet cryptography/SDK, and ZK circuits. Reports (and re-audits after material changes) are published on the transparency page.

Q16: How are operators (provers/relayers) kept honest?

They stake $NOC to register. Misbehavior (invalid proofs, liveness failures, fraud) can be slashed, with proceeds redistributed/burned per DAO policy.

Q17: What data does Noctura store on-chain?

Only public artifacts needed for verification: commitments, nullifiers, and the Merkle root. No private notes/PII are written on-chain.

Q18: Can enterprises use this without breaking compliance?

Yes. The wallet supports policy controls (dual-control, limits), scoped view access for finance/compliance, and audit tokens for regulators/partners.

Q19: What's on the near-term roadmap?

Stage 1 (2025 Q4): On-chain presale + Wallet Beta on testnet
Then: devnet shielded pools, testnet verifiers, **shielded swaps beta**, mainnet shielded layer, listings, DAO bootstrap, and scale-out via proof aggregation.

Q20: How are treasury and parameters governed?

By the DAO with multisig + timelock rails. Voters control emission curves, fee-burn %, operator min-stake, batch windows, and circuit activations; safety caps/floors and emergency pause are hard-coded.

Q21: Where can I see transparency data?

The transparency page publishes contract addresses, audits, restricted-jurisdiction policy, DAO parameters, quarterly emissions/fees/burn reports, and changelogs for circuits/parameters.

Q22: What are the main risks?

Regulatory changes, cryptographic/implementation bugs, market volatility, and operator centralization. Mitigations include audits, bug bounties, guarded upgrades, staking/slashing, and conservative throughput targets.

Q23: How do I build on Noctura?

Use the Wallet SDK (transparent & shielded ops, cross-mode transfers, disclosure APIs). See Section 11 for schemas, proof formats, verifier interface, and example flows.

Q24: Support & security contact?

Security disclosures via PGP email (listed on the transparency page). General support via docs, Discord/Telegram, and issue trackers linked from the website.

Q25: How do private swaps work?

Noctura's shielded swap feature lets you exchange tokens privately within the wallet. Unlike public DEX swaps that reveal your trades, positions, and strategies, shielded swaps use zk-proofs to hide:

- Trading pairs (what you're swapping)
- Amounts (how much you're trading)
- Balances (your wallet holdings)
- Timing patterns (when you trade)

Swaps are executed via private liquidity pools with fees paid in NOC ($0.006-$0.008 flat rate) to prevent fee-amount correlation attacks.

Q26: Why are fees so low ($0.0005-$0.008)?

Noctura uses flat-rate fees for two reasons:

1. **Privacy:** Amount-based fees leak transaction size. Flat fees prevent this correlation
2. **Efficiency:** Off-chain proving + on-chain verification keeps costs minimal (~150-300k CU)

The fee covers computational costs (proof verification, state updates) and compensates the prover network. Flat pricing makes privacy accessible for everyday transactions.

Q27: Why must fees be paid in NOC instead of SOL or USDC?

NOC is technically required for privacy—it's not an arbitrary choice:

- **Fee visibility:** Paying fees in the same token being transferred (e.g., SOL fees for private SOL sends) would reveal balance changes
- **Multi-token problem:** Supporting different fee tokens fragments liquidity and enables correlation attacks based on fee token choice
- **Unified layer:** A single privacy-native fee token (NOC) creates uniform cost structure across all operations, preventing fee-based deanonymization

NOC is the cryptoeconomic primitive that makes the shielded layer work—it's infrastructure, not a tax.

Q28: What's the difference in fees for transfers vs swaps?

- **Private transfers:** $0.0005 NOC (simple send, ~150k CU)
- **Private swaps:** $0.006-$0.008 NOC (multi-token proof, routing, ~250-300k CU)
- **Cross-mode ops:** $0.001-$0.003 NOC (deposit/withdraw with unlinkability)

All fees are flat-rate regardless of amount to preserve privacy.

15. Governance

15.1 Phased Governance (Core → DAO)

Phase 0 — Launch Controls (pre-mainnet & presale)

- Multisig (core contributors + independent advisors) manages: presale program, token mint, audit sign-offs, and non-contentious ops (e.g., transparency page)
- Scope intentionally narrow; no discretionary treasury moves beyond budget rails

Phase 1 — Guarded Parameters (mainnet TGE → stability)

- Multisig + timelock administer protocol parameters within hard guardrails set on-chain (caps/floors)
- Community signaling via off-chain snapshot; non-critical changes (e.g., fee-burn %, batch windows, operator min-stake) require timelocked execution

Phase 2 — DAO Control (post-audit, stable ops)

- On-chain DAO assumes control of: emissions schedule knobs, operator policy, fee-split, treasury grants, and circuit activation after audits
- Multisig remains as executor of DAO-passed proposals only; no unilateral authority

Phase 3 — Mature Decentralization

- DAO governed end-to-end with constitutional guardrails (immutables): emergency pause scope, max issuance rate, circuit security invariants, and disclosure API baselines

15.2 Use of Proceeds

Noctura's treasury is deployed to accelerate delivery of the dual-mode wallet, harden security, grow ecosystem adoption, and ensure deep, healthy liquidity for launch and expansion. All treasury movements follow governance controls (e.g., multisig + timelock) and are executed within predefined budget rails, with periodic transparency reporting.

Allocation Breakdown

1. **Liquidity provisioning (DEX/CEX depth; not for price support) — 22%**
   Used for initial liquidity deployment, market-making tooling, and exchange liquidity requirements where applicable. This allocation is strictly for liquidity health and operational readiness—not artificial price support.
2. **Security, audits & bug bounty — 18%**
   External audits across wallet code, smart contracts/programs, zk circuits/verifier logic, and infrastructure. Includes an ongoing bug bounty reserve.

3. **Core engineering & infrastructure — 20%**
   Wallet development (Transparent + Shielded UX), performance engineering, prover/relayer lanes, monitoring, redundancy, and production-grade reliability.
4. **Integrations & SDK adoption — 12%**
   Developer SDKs, documentation, reference integrations, partner engineering, and tooling to enable dApps and wallets to integrate Noctura privacy primitives safely.
5. **Ecosystem grants — 8%**
   Grants to teams building privacy-enabled Solana apps, integrations, analytics that preserve privacy, and community tooling that expands the Noctura ecosystem.
6. **Marketing & partnerships — 10%**
   Growth campaigns, partnerships, community expansion, and strategic distribution to drive adoption and awareness through measurable, trackable channels.
7. **Legal & compliance operations — 5%**
   Legal structuring, policy work, restricted-jurisdiction controls, and compliance adapter support as required by partners and listing venues.
8. **Reserve / contingency — 5%**
   Buffer for unforeseen events, security incidents, critical upgrades, and operational continuity, deployed only under governance controls.

Flexibility & Controls

To remain adaptive, allocations may shift modestly based on real-world needs (e.g., audit scope expansion, integration demand, listing requirements). Any material reallocation is executed under governance controls and recorded in treasury reporting.

15.3 Voting Power (Staked $NOC), Quorums, Safeguards

Voting Power

- 1 staked $NOC = 1 vote (at proposal snapshot)
- Optional longer-lock multiplier (bounded, e.g., up to 1.5×) to reward long-term alignment without plutocracy

Quorum & Thresholds

- **Standard proposals:** quorum ≥ 10% of staked supply; simple majority to pass
- **Sensitive proposals** (emissions curve, fee-burn %, operator min-stake): quorum ≥ 15%; 60% supermajority
- **Critical proposals** (circuit activation, verifier upgrade, emergency powers): quorum ≥ 20%; 66% supermajority

Safeguards

- **Timelock** (e.g., 72h–7d) on all state-changing executions; immutable minimum enforced in contracts
- **Parameter guardrails:** caps/floors embedded on-chain (e.g., max fee-burn %, max emission rate, min operator stake)

- **Conflict checks:** proposals auto-rejected if violating invariants (e.g., spend > treasury, emission > cap)
- **Sybil/flash-loan resistance:** snapshot voting (block-height), stake lock requirement during voting window

Transparency

Public proposal texts, diffs, on-chain simulations, and risk notes; all executed actions linked to proposal IDs.

15.4 Upgrade Process & Emergency Controls

Upgradeable Components

- Verifier program (proof verification, tree ops)
- Circuit registry (vk ids, circuit versions)
- Operator/treasury modules (staking, fee routing)

Standard Upgrade Pipeline

1. Spec & audit published (diff, test vectors)
2. DAO vote approves upgrade; includes parameter diffs and activation conditions
3. Timelock starts; binaries/IDLs pinned (IPFS/commit hash)
4. Shadow deploy to canary/testnet; monitoring window
5. Activation via DAO executor; automatic rollback plan defined

Circuit Activation

- New circuits (vk ids) registered alongside current versions; dual-acceptance window permits safe migration
- Deactivation of old circuits only after cutover quorum (usage threshold) or time-based sunset, whichever first

Emergency Controls (narrowly scoped)

- **Emergency Pause (shielded submit only):** Freezes new shielded transactions while preserving read access, staking accrual, and claims
- **Triggers:** Verified critical vulnerability, integrity loss (e.g., proof forgery), or exploit in progress
- **Process:** 2-of-N emergency council (multisig subset) → immediate pause; mandatory DAO ratification within a fixed window (e.g., 7 days) or auto-unpause
- **User safety:** Funds remain accessible; exits via transparent paths stay available if not implicated

Post-incident

- Public post-mortem with timelines, root cause, and remediation
- Re-audit required for affected modules; resume via timelocked DAO approval

Immutable Invariants

Cannot be changed by any vote: max supply, nullifier uniqueness rule, conservation constraint (Σinputs = Σoutputs + fees), minimum timelock floor, and disclosure API consent requirement.

Outcome: Governance balances agility (shipping upgrades, responding to incidents) with credible neutrality (DAO control, hard guardrails, and transparent processes) so the protocol can evolve safely without compromising user trust.

16. Risks & Disclosures

16.1 Regulatory, Technical, and Market Risks

Regulatory

- **Policy shifts:** Privacy tooling may face new restrictions, licensing, reporting, or geo-fencing mandates. Jurisdictional divergence can limit features or access
- **Exchange posture:** Listings and fiat ramps may require additional attestations (view keys, audit tokens, KYC pointers). Loss of listings/liquidity is possible
- **Sanctions/blacklists:** Counterparty controls and Travel-Rule obligations can change with little notice; adapters may be disabled regionally

Technical

- **Cryptographic assumptions:** Soundness relies on hardness of chosen curves/hashes and SNARK security. A break or implementation flaw could compromise privacy or integrity
- **Implementation bugs:** Verifier logic, circuit constraints, Merkle/tree updates, or wallet key management could contain defects despite audits and testing
- **Throughput/latency limits:** Shielded TPS and confirmation times depend on proof generation, on-chain verification, and network load; targets (hundreds TPS) are not guarantees
- **Operator centralization:** Prover/relayer concentration may degrade censorship resistance or quality of service if not countered by incentives and staking policy
- **Upgrades & compatibility:** Circuit/version migrations may cause temporary disruptions; clients must update to maintain functionality

Market

- **Price volatility:** $NOC may fluctuate significantly due to market conditions, liquidity depth, or sentiment
- **Adoption uncertainty:** User/partner uptake of shielded flows and disclosure APIs may be lower or slower than expected
- **Competition:** Alternative privacy solutions or L1/L2 roadmap changes could reduce Noctura's relative advantage

16.2 Operational & Treasury Risks

- **Treasury management:** Misallocation, custodial failures, or market drawdowns can reduce runway. Treasury actions are timelocked and public but cannot eliminate loss risk
- **Liquidity provisioning:** Initial pool depth and market-maker performance may be insufficient, causing slippage and volatility
- **Key management & governance:** Multisig/DAO keys may be compromised through phishing, device loss, or collusion. Safeguards include role separation, hardware keys, timelocks, and immutable guardrails
- **Third-party dependencies:** RPC providers, KYC vendors, and Travel-Rule partners can introduce outages or policy changes that affect UX or availability
- **Incident response:** Emergency pause halts new shielded submits but may not prevent all loss modes. Recovery may require contract upgrades subject to governance and re-audit

16.3 Investor Disclaimer (No Guaranteed Yields)

- **No promises of profit:** $NOC is a utility token for fees, staking, governance, and operator incentives within the Noctura ecosystem. It is not an investment contract, deposit, or security claim on protocol revenues
- **Variable rewards:** Staking APR bands (e.g., 128%/365d, 68%/182d, 34%/90d) and any time-boxed presale boosts are targets, not guarantees. Actual returns vary with total staked, emissions schedules, protocol fees, and DAO decisions
- **Risk of total loss:** Token value can go to zero. Smart-contract or cryptographic failures, regulatory bans, market crashes, or operational incidents can result in partial or total loss of funds
- **Jurisdictional limits:** Access to features (shielded mode, presale, staking) may be restricted by location, KYC status, or partner policy. Users are responsible for complying with local laws and tax obligations
- **Forward-looking statements:** Roadmap items, throughput targets, listings, and integrations are goals that may change without notice based on audits, feasibility, and governance outcomes

Proceed only if you fully understand and accept these risks. Always use hardware-secured wallets, and never commit more than you can afford to lose.

17. Appendices

A. Glossary & Notation

- **Anchor / Root (R)** — Current Merkle tree root for shielded commitments used as the public reference in proofs
- **Commitment (C)** — Hash binding a note's contents (recipient key, mint, amount, randomness)
- **Nullifier (N)** — One-time, unlinkable marker proving a note was spent (prevents double-spend)
- **Note** — Private record of value in the shielded set
- **Spend Proof** — ZK proof showing membership, authorization, balance conservation, and uniqueness of nullifiers
- **Verifier Program** — Solana on-chain program that validates proofs and updates state (R, C[], N[])
- **View Key** — Read-only key granting scoped viewing (tx/time/balance) without spend authority
- **Audit Token** — Consent-bound, expiring assertion (e.g., proof-of-funds, KYC pointer) verifiable by partners
- **Operator (Prover/Relayer)** — A staked service that generates proofs / relays shielded txs; subject to slashing
- **DAO** — On-chain governance controlling parameters (emissions, fee-burn %, operator policy)
- **TGE** — Token Generation Event

B. Cryptographic Parameters (Curves, Hashes, Tree Depths)

Parameters are versioned. Final values are pinned in the verifier registry and audit reports.

- **Proof System (launch):** zk-SNARK (Groth16/PLONK family) on BLS12-381
- **Hash Function (commitments / Merkle):** Poseidon (field-friendly), domain-separated for C, N, and tree nodes
- **PRF for Nullifiers:** Poseidon-based PRF keyed by sk_spend with explicit domain tags
- **Merkle Tree:**
    - Depth: 32 (launch) with upgrade path to 40/48 via versioned trees
    - Arity: Binary
    - Witness Format: sibling hashes + bit-indices (LSB first)
- **Range Proofs:** Amount range constrained in-circuit (non-negative, cap < 2^64) with overflow checks
- **Transcript Binding:** Fiat-Shamir with domain tags for spend, deposit, withdraw circuits
- **Key Derivation:** Ed25519 (transparent), shielded spend/view keys from master seed via hardened paths; disclosure branch separated

C. Economic Parameters (Stage Prices, Emission Caps)

- **Total Supply:** 256,000,000 $NOC (fixed)
- **Presale Allocation:** 102,400,000 (40%), sold in 10 equal stages (10,240,000 each)
- **Price Ladder** (4-dec; exact hard cap $25,600,000):

Staking Emissions Pool: 51,200,000 $NOC (20% of supply)
Post-TGE Staking APR Bands: 128% (365d), 68% (182d), 34% (90d); DAO-tunable within capped ranges
Operator Rewards: Share of protocol fees/emissions split per delivered QoS; DAO-tunable
Optional Fee-Burn: 0.25–1.00% of shielded fees (DAO-set), burned prior to reward splits

D. Contract Addresses & Registry

All addresses are published on the transparency page and versioned in a registry account. Only the presale address is final at this time.

- **Presale Program (Mainnet):** `6nTTJwtDuxjv8C1JMsajYQapmPAGrC3QF1w5nu9LXJvt`
- **$NOC SPL Mint:** TBA (post-audit, pre-TGE)
- **Verifier Program (Shielded):** TBA (testnet → mainnet after audits)
- **Commitment Tree Account(s):** TBA
- **Nullifier Set Account(s):** TBA
- **Treasury / Multisig:** TBA (DAO bootstrap stage)
- **Registry Account (pins current versions):** TBA

Change control: New deployments are added; old ones marked deprecated only after cutover. Each entry includes program id, commit hash/IPFS ref, audit link, and activation slot.

E. Example JSON Schemas (API/Proof Objects)

Canonical schemas live in the SDK repo; below are abridged, human-readable versions for integrators.

E.1 Spend Proof Envelope

```
{
  "schema_v": "spend.v1",
  "vk_id": "0x<32>",
  "anchor_slot": 259921430,
  "merkle_root": "0x<32>",
  "nullifiers": ["0x<32>", "0x<32>"],
  "commitments": ["0x<32>", "0x<32>"],
  "fee_commitment": "0x<32>",
  "memo_commitment": "0x<32>",
  "proof": "0x<bytes>"
}
```

E.2 Merkle Path Request/Response

Request:


```
{ "C": "0x<32>", "root": "0x<32>" }
```


Response:


```
{
  "path": ["0x<32>", "..."],
  "indices": [0,1,1,0, "..."],
  "depth": 32
}
```


E.3 View Key & Audit Token

View Key (scoped):


```
{
  "schema_v": "viewkey.v1",
  "id": "vk_01H...",
  "scope": { "kind": "time_window", "from": 1700000000, "to":
1700600000 },
  "key": "base64...",
  "expiry": 1701200000
}
```

Audit Token (consent-bound assertions):

```
{
  "schema_v": "audit.v1",
  "token_id": "at_01H...",
  "auditor_pk": "0x<32>",
  "scope": { "kind": "proof_of_funds", "mint": "So111...", "amount":
"10000" },
  "zk_assertions": [
    { "kind": "kyc_pointer", "hash": "0x<32>" },
    { "kind": "origin_clean", "policy": "v1.2" }
  ],
  "ttl": 86400,
  "sig_user": "0x<64>"
}
```

E.4 Wallet Operations (abridged)

Shielded Send:

```
{
  "op": "sendShieldedTx",
  "req": {
    "outputs": [
      { "pk_shielded": "0x<pk>", "mint": "So111...", "amount": "250" }
    ],
    "feeMint": "NOC",
    "anchor": "0x<32>",
    "prover": "lane:gpul1",
    "maxLatencySec": 3
  }
}
```

Cross-Mode Transfer:

```
{
  "op": "crossModeTransfer",
  "req": {
    "direction": "public_to_shielded",
    "mint": "So111...",
    "amount": "1000"
  }
}
```

Errors (enum):

```
{
  "error": "ANCHOR_STALE",
  "details": "Provided anchor 0x... is older than allowed epoch window"
}
```

These appendices provide the pinned vocabulary, cryptographic and economic knobs, canonical addresses, and integration schemas needed to build, audit, and operate against Noctura's shielded privacy layer on Solana. As components graduate from testnet to mainnet, the registry and transparency page will be updated with final IDs and audit links.

18. Call to Action

Privacy that survives the real world starts here. Noctura delivers a shielded privacy layer on Solana with a dual-mode wallet, private swaps, and selective disclosure—usable for everyone, acceptable to partners, and verifiable on-chain.

Start Now

- **Join the presale:** noc-tura.io/presale
- **Claim at TGE** & optionally auto-stake into your chosen lock (128%/365d, 68%/182d, 34%/90d)
- **Test the Wallet Beta (testnet):** Try Transparent ⇄ Shielded, deposits/withdrawals, private swaps, and View Keys

Build With Us

- **Developers:** Integrate private settlement, swaps, and scoped audits via the Noctura Wallet SDK (Section 11)
- **Operators:** Apply to run prover/relayer lanes (stake $NOC, earn fees; slashing-secured)
- **Enterprises:** Enable policy controls (dual-control, limits) and Audit Tokens for compliance workflows

Stay Aligned

- Track audits, parameters, and addresses on the **Transparency Page**
- Participate in **DAO votes** (emissions, fee-burn %, operator policy) once governance opens
- Join the **community channels** for releases, bounty calls, and partner programs

Choose privacy—without losing legitimacy.

Noctura

Shielded privacy layer on Solana, powered by a dual-mode wallet and the $NOC economy.

Email: privacy@noc-tura.io
Website: www.noc-tura.io

Token Address: B61SyRxF2b8JwSLZHgEUF6rtn6NUikkrK1EMEgP6nhXW
Presale Contract: 6nTTJwtDuxjv8C1JMsajYQapmPAGrC3QF1w5nu9LXJvt