

Shadow Swap – Encrypted Orderflow DEX on Solana

Project Overview

Shadow Swap is a privacy-preserving decentralized exchange (DEX) on Solana that uses Arcium's encrypted compute layer and private execution routing via Sanctum Gateway and Jito bundles. It aims to protect traders from MEV (maximal extractable value) attacks, ensure reliable transaction inclusion, and enable institutional-grade private execution for both individuals and whales.

What Problem Are We Solving?

1. MEV / Front-Running Exposure

- Traditional DEXs expose order details in public mempools.
- Bots monitor the mempool and front-run or sandwich user transactions.
- Large orders suffer from slippage and alpha leakage.

2. Execution Reliability

- Solana transactions often fail due to congestion.
- Users lose fees and miss trading opportunities.

3. Whale + Institutional Trading Constraints

- Large traders want to hide intent, pricing, and positions.
- Lack of confidentiality limits serious institutional adoption.

Core Innovations

✓ Encrypted Orderflow Matching (Arcium)

- Orders are encrypted client-side and matched privately using MPC.
- Neither Arcium nodes nor validators can see order parameters until execution.
- Matching is deterministic and trustless.

✓ Atomic Execution with Private Delivery

- Matched trade is decrypted (threshold MPC), and transaction built.
- Delivered privately to validators using Jito bundle via Sanctum Gateway.
- Eliminates exposure in Solana public mempool.

✓ Institutional-Ready Privacy Stack

- Hidden order book, MEV-resilient execution, and guaranteed settlement.

- Optional fallback via RPC disabled for sensitive orders.

System Architecture

Components:

- Frontend: Next.js + Arcium SDK + Solana Wallet Adapter
- Backend / MPC: Arcium Cluster (MXE)
- Settlement Program: Solana smart contract (Anchor)
- Execution Router: Sanctum Gateway (private first)
- Transaction Path: Jito bundle → Validator → On-chain settlement

Data Flow:

graph TD;

A[User Order UI] --> B[Encrypt Order via Arcium SDK];

B --> C[Send to Solana Contract];

C --> D[Queue to Arcium MXE];

D --> E[MPC Matching];

E --> F[Threshold Decryption];

F --> G[Generate Execution Tx];





G --> H[Send via Jito Bundle via Sanctum];

H --> I[Validator Settlement];

I --> J[Trade Finalized On-Chain];

Features Recap

User Experience

-  Encrypted order book (no public visibility)
-  Zero MEV leakage (even post-trade info protected)
-  99.9% execution reliability
-  Real-time analytics dashboard (transaction success, MEV saved)

Trading Logic

- All orders submitted via Arcium SDK (X25519 + Rescue cipher)
- Match engine runs inside MPC cluster (both buy/sell sides encrypted)
- Final trade result passed to Solana via threshold-decrypted callback
- Callback function performs asset transfers via escrow PDAs

Trade Execution

- Developer builds the Solana transaction post-match
- Sanctum Gateway routes via Jito private bundle
- No fallback to public RPC unless explicitly enabled (disabled by default)

Optional: Public Liquidity Routing (Jupiter)

- If no private match is found, fallback logic can route to Jupiter
- NOTE: This is not built into Arcium. Developer must implement:
 - Off-chain price quoting via Jupiter
 - Fallback match logic in Arcium's MPC
 - Callback handler that calls Jupiter program on-chain



Development Workflow

Phase 1: Core Integration (Arcium + Sanctum)

- Encrypt + Submit Orders
- Arcium order queueing & matching logic
- Callback handler to process match result
- Smart contract asset transfer via PDAs
- Build Solana TX from MPC output
- Sanctum Gateway integration → Jito bundles

Phase 2: Execution Control & Privacy Guarantee

- Disable fallback to public RPC by default
- Tune tip priority for Jito validators
- Add analytics for match vs. fallback execution path
- Show event log for user (post-trade only)

Phase 3: Optional Jupiter Integration

- Route unmatched orders to public liquidity (optional)
- Encrypt route quote + integrate into MPC match logic
- Submit fallback tx to Jupiter program



Summary: Why This Architecture?

Goal	How We Achieve It
Private matching	Arcium MPC + X25519 order encryption
Zero MEV risk	Threshold decrypt → Jito bundle delivery only
Reliable settlement	Sanctum Gateway with multipath retries
Large trade privacy	No mempool leaks, even on fallback
Institutional trust	No signature post-match, trustless callback flow

What Arcium Handles vs What We Build

Stage	Arcium Provides	We Implement
Encryption	SDK (X25519, Rescue)	Client logic to use SDK
MPC Matching	MXE cluster infra	Write custom matching logic
Result delivery	Callback trigger + small data delivery	On-chain Anchor handler
Execution	None	We build TX + route via Sanctum/Jito
Public Liquidity	Not supported	Build Jupiter fallback (optional)

Configuration Notes

- All order submissions encrypted via MXE public key
- No orders ever hit public mempool unless explicitly routed
- Sanctum Gateway configured to JITO_ONLY for sensitive flows
- Arcium callback server used only for large outputs (optional)

User Roles

- Trader A: Places private buy/sell intent via encrypted UI
- Trader B: Counterparty via same platform (order also encrypted)
- Protocol Owner (you): Runs Arcium-integrated smart contract and defines matching rules + fallback behavior

Appendices

- Arcium Cluster Key Management
- Solana PDA design for escrowed asset control
- Arcium Callback Server handler (for large MPC outputs)
- Fallback policy logic (private_only, fallback_after_n_sec, etc.)