Post-Quantum Non-Custodial Brokerage

PQSE Research Team

quantaswap@gmail.com

July 2025

Abstract

We propose a non-custodial brokerage protocol designed to enable decentralized trading of financial instruments under adversarial, post-quantum conditions. The system integrates conventional limit-order execution with automated market making (AMM), enabling hybrid liquidity provisioning without dependence on centralized intermediaries. A referral-incentivized fee mechanism ensures market sustainability while preserving execution determinism. All balances and token flows are 64-bit bounded to prevent overflow and ensure cross-implementation consistency.

PQSE—Post-Quantum Securities Exchange—executes trades deterministically using on-chain matching logic, supporting best-effort netting across bidirectional flows. Orders remain live for a minimum enforced duration, protecting market integrity from adversarial churn. The exchange architecture is intentionally hostile to price manipulation, liquidity griefing, and centralized rent extraction. All inputs and outputs are auditable and non-custodial, while trading fees are discountable via a utility token and redistributable through an on-chain dividend model.

This paper outlines the system architecture, core protocol mechanics, and adversarial assumptions under which PQSE operates. We further provide the mathematical model governing execution price limits, matching priority, and liquidity sharing across orderbook and pool mechanisms.

Keywords: decentralized exchange, non-custodial, post-quantum, limit orders, hybrid liquidity, deterministic execution, IZRC20, on-chain settlement

1. Introduction

Centralized exchanges have historically acted as the default brokerage mechanism for digital asset trading. While efficient under ideal conditions, these systems introduce structural dependencies on custodial control, operational opacity, and compliance bottlenecks. Worse, they remain fundamentally brittle under quantum-capable adversaries, particularly where digital signature schemes are non-hardened.

We present a protocol—PQSE—that removes the need for trusted third parties in the context of order-driven trading. The system executes deterministic trades across a limit-order book and constant-product market maker, using only on-chain logic and user-signed inputs. Market participants retain full custody of their assets until the moment of trade finalization. The architecture permits multiple in-flight intents per actor and enables pro-rata netting without favoring latency arbitrage or gas-price gamesmanship.

All trades occur within a finite precision domain: token balances, prices, and fees are explicitly 64-bit bounded. This constraint is enforced contractually and numerically, allowing for interoperable execution across a wide range of computing environments and reducing the likelihood of edge-case inconsistencies across protocol clients.

The system assumes adversaries with read and front-run capabilities but no private-key compromise of FREE token holders. Discounted trade fees are computed deterministically using fixed-point arithmetic and are rebated via a utility token (FREE). Protocol revenue is distributed pro-rata to holders of this utility token, forming a closed-loop feedback incentive without relying on governance-based fee routing.

PQSE does not aim to maximize liquidity or fee extraction. Its primary goal is fairness under constraint—ensuring that users with symmetrical intents receive symmetrical outcomes, irrespective of off-chain bargaining power or infrastructure latency.

1.1 Design Constraints

The design of PQSE is grounded in the following hard assumptions:

- **Adversarial model:** All mempool activity is observable; transaction reordering is assumed.
- **Execution determinism:** No part of the trade path may branch based on timing or randomness.
- **Finite-width accounting:** Token balances, fees, and liquidity shares operate within 64-bit unsigned integers.
- **Quantum resilience:** All custodial pathways must be avoidable or hardened against post-quantum cryptanalysis.
- **Censorship resistance:** No step of trade execution requires external approval or interactive coordination.

In the sections that follow, we outline the protocol's order semantics, trade-matching algorithms, liquidity accounting, and the mechanisms through which protocol revenue is accrued and distributed.

2. Protocol Architecture

PQSE is structured as a hybrid execution environment combining deterministic limit-order matching with fallback access to automated market making (AMM). Both pathways are integrated into a unified matching pipeline that operates entirely on-chain and requires no off-chain sequencer or coordinator.

2.1 Token Model

All assets on PQSE conform to the `IZRC20` standard, which extends ERC-20 with:

- 64-bit supply and balance domains (`uint64`).
- Explicit support flags for token mobility and custodianship delegation.
- Optional batch transfer interfaces for gas optimization.

Tokens that fail `IZRC20` compliance checks are rejected at pair creation and during any trade or liquidity operation.

2.2 Pool Structure

Each token pair (R, S) defines a unique constant-product AMM pool:

- R: reserve asset, used for fee accounting.
- S: secured asset, never held by the protocol except during trades.

The pool maintains:

$$\begin{aligned} R_0 &\in N_{64} \quad \text{(reserve balance)} \\ S_0 &\in N_{64} \quad \text{(secured balance)} \\ k &= R_0 \cdot S_0 \quad \text{(invariant)} \\ \texttt{shares} &\in N_{128} \quad \text{(LP ownership state)} \end{aligned}$$

Liquidity providers are issued internal IDs (`loc`) that encode their proportional claim on the pool. Fee rewards are paid to FREE token holders, not LPs.

2.3 Limit Orders

PQSE uses explicit limit orders, not passive liquidity positions. Each order defines a directional trade intent with price bounds:

- $r > 0$: offer reserve to buy secured when *cheap*.
- $s > 0$: offer secured to buy reserve when *expensive*.
- $[t_{\min}, t_{\max}]$: tick window defining price range.
- $t_{\exp}$: optional expiry timestamp.

If both sides are funded, the order acts as a symmetric **"buy low, sell high" command**—it will buy secured below a defined price and sell it above another. Orders are fully funded at submission and do not simulate virtual liquidity between ticks.

This mechanism differs fundamentally from Uniswap v3: PQSE does *not* interpolate liquidity across a tick range. There is no passive exposure. Every order has discrete, bounded intent and executes strictly if-and-only-if its price and direction are matched.

2.4 Brokers and Delegation

Any address may authorize a broker to act on its behalf with an expiry timestamp:

```
approve(broker, expires)
```

Brokers may submit trades, cancel orders, or manage liquidity positions within the defined time window. Delegation is opt-in and non-renewing by default.

2.5 FREE Token Mechanics

The FREE token governs fee discounts and profit redistribution:

- Holding up to 1 full unit yields a linear discount from 0–100%.
- The fee is split into: protocol share, liquidity rebate, referral tip.
- Protocol share is redistributed pro-rata to FREE holders on-demand.

Discount calculations and referral payouts are deterministic and bounded. There is no off-chain governance process or dynamic fee tuning.

2.6 Execution Path

Trade execution is deterministic, adversarial-aware, and pipeline-driven:

1. User submits a `Cross` intent including token inputs, limit orders, and strike guardrails.
2. The engine:
   (a) Sweeps the orderbook in price-time order until it reaches pool price.
   (b) Nets opposing flows directly at current pool price.
   (c) Executes remaining flow via a "slam" pass: alternating book and pool until exhaustion.
3. All inputs are distributed pro-rata to outputs. Any unused flow is returned.
4. Fees are applied before and after execution. Execution price is validated against the strike corridor.

All routing logic is encoded in contract code. There is no gas auction, no searcher dependency, and no sequencing required.

3. Trade Semantics

All trading in PQSE is initiated via an explicit, structured submission called a `Cross`. This object encodes the full trading intent of a multi-party transaction and is processed atomically. It includes limit order insertions, liquidity contributions, and cross-side inputs.

3.1 Cross Structure

A `Cross` contains the following components:

- **Pair:** token pair (R, S)
- **Makers:** array of new limit orders, grouped by maker
- **Patches:** optional order slot bindings for later use
- **Takers:** array of existing order IDs to be consumed
- **Inputs:** side-specific token deposits from users or brokers
- **Orders:** allocation slots to record which user funds which flow
- **Guardrails:** min/max strike ticks enforced post-execution

This structure generalizes intent batching: multiple actors may fund either side of a trade, add new orders, or consume old ones. The matching engine operates over the entire structure in a single pass.

3.2 Funding Semantics

Users may contribute tokens to either the reserve or secured side of a trade. Each input is accompanied by a declared owner and an amount. Ownership is verified at runtime; if the sender is not the owner, then broker delegation must be in force and unexpired.

Inputs are aggregated and staged for matching. No tokens are spent unless execution conditions are met.

3.3 Matching Pipeline

Matching proceeds in four deterministic stages:

1. **Book Sweep:** Traverse each side's order book in price-time order until the opposing flow is depleted or pool price is reached.
2. **Internal Netting:** Cross-match remaining reserve and secured inputs at current pool price. No pool liquidity is touched.
3. **Final Slam:** Residual flows are alternately routed into the book and pool until exhausted or no viable trades remain.
4. **Justification:** Outputs are distributed to original contributors based on pro-rata input weighting.

All steps are gas-bounded and halt once convergence is achieved. No further optimization or dynamic routing is attempted.

3.4 Order Patching and Takers

Makers may provide optional `Patch` structs indicating which order IDs should be overwritten or aliased. This allows flexible coordination between traders without forcing upfront coordination.

Takers may specify an array of order IDs to cancel. If the caller is not the order's original owner, broker rights must be in force. Cancelling an order refunds any unspent balances (at most two transfers: one per side).

3.5 Execution Boundaries

After execution, the contract computes the realized execution price in both directions:

- Buy-side: $P_{\text{buy}} = \dfrac{\texttt{securedOut}}{\texttt{reserveIn}}$
- Sell-side: $P_{\text{sell}} = \dfrac{\texttt{securedIn}}{\texttt{reserveOut}}$

Both values are compared to the user's declared strike corridor. If either side's realized price falls outside its respective bound, the transaction reverts.

3.6 Determinism and Replayability

Every `Cross` is self-contained and produces deterministic output given chain state. Execution proceeds identically across nodes and cannot be front-run into different behavior. No state randomness or latency-based priority exists. Execution paths are structurally invariant.

4. Security Model

PQSE is designed under the assumption that adversaries possess full mempool visibility, network timing advantages, and zero-cost read access to all public chain state. The protocol does not rely on optimistic sequencing, probabilistic ordering, or off-chain precommitments. Security is enforced entirely by contract structure and execution determinism.

4.1 Threat Assumptions

We assume the following attacker capabilities:

- **Mempool Monitoring:** Adversaries observe all transactions prior to inclusion and may reorder them using bribes or private relays.
- **Gas Priority:** Adversaries may front-run, back-run, or insert intermediate transactions at will.
- **Signature Breaking:** Quantum adversaries may break classical digital signature schemes used by ERC-20s.

- **Orderbook Sniping:** Adversaries monitor tick ranges and attack narrow-band orders through slippage exploitation.
- **Liquidity Griefing:** Adversaries may flood the book with one-sided orders or time-limited spam.

The protocol remains correct and fair under these conditions. Liveness is dependent on block inclusion but correctness is deterministic.

4.2 Mitigation Strategies

PQSE resists adversarial behavior through structure and finite-state safety:

- **Best Execution:** Orders are matched in price-time order; no selective skipping or fee-based priority exists.
- **Non-custodial Invariants:** Tokens remain under user control until trade finalization. No custodial holding is required.
- **Expiry Enforcement:** Orders may include hard expiry times; minimum durations are enforced to prevent ephemeral spam.
- **Deterministic Matching:** Matching engine behavior is fully deterministic. Identical input always yields identical output.
- **Strike Corridors:** Traders may specify price bounds. Any deviation outside this corridor causes the transaction to revert.
- **64-bit Bounding:** All token, fee, and price fields are bounded to 64-bit integers. No unbounded multiplication occurs.

Fee revenue distribution is also deterministic and cannot be redirected by governance. There are no centralized admin keys.

4.3 Post-Quantum Considerations

All user custody is implemented using IZRC20 tokens, which must support key-rotation and/or post-quantum signature schemes. PQSE does not mandate any specific cryptographic backend but assumes that token implementations are quantum-resilient.

FREE token usage is non-custodial and requires no signature reuse. Revenue claims may be made in arbitrary order and are idempotent.

4.4 Griefing and Fairness

PQSE ensures execution fairness under congestion. There are no gas auctions, no auction-based batchers, and no matching delays. Orderbook integrity is protected through minimal live-time requirements and strictly bounded execution gas. Slippage is controllable via user-supplied price corridors.

No user can be forced to transact at a worse-than-declared price.

4.5 Failure Modes

PQSE does not halt on partial failure. The following conditions are handled gracefully:

- **Token Transfer Failure:** If any input token transfer fails, the participant is skipped; the transaction proceeds with remaining inputs.
- **Orderbook Underflow:** If all orderbook orders are exhausted or crossed, execution defaults to internal swap and AMM fallback.
- **Fee Overflow:** If fee calculation overflows 64-bit domain, the trade is rejected.
- **Price Guard Violation:** If realized price exits user corridor, the entire trade is reverted.

These outcomes are structural, not emergent, and never result in asset loss.

5. Liquidity Mechanics

PQSE supports constant-product automated market making (AMM) as a liquidity fallback mechanism for cases where the limit order book is empty or underpriced. The AMM is deterministic, price-bounded, and 64-bit safe. Liquidity providers (LPs) interact through internal position IDs and receive no protocol fees.

5.1 AMM Function

Each pool (R, S) maintains a constant-product invariant:

$$k = R \cdot S$$

where R and S are the reserve and secured token balances, respectively. The AMM offers strictly convex pricing. All swaps are slippage-bounded and calculated in fixed-point arithmetic (Q64.64 domain).

The AMM is used only when:

- No matching limit orders remain at better price.
- Internal netting is incomplete.
- The caller explicitly accepts potential pool slippage.

The AMM is not the primary liquidity source. It is an execution fallback and price stabilizer.

5.2 Liquidity Provision

Liquidity may be added via two functions:

- `initializeLiquidity()`: bootstrap a new pool (first LP only).
- `depositLiquidity()`: add liquidity to an existing position or create a new one.

Initial LPs must provide strictly positive and balanced token amounts. The first LP position must exceed a protocol-defined threshold:

$$\min(R_0, S_0) \geq \texttt{MIN\_LIQ}$$

All positions are tracked via internal IDs (‘loc‘) and balances are credited to the pool. LP tokens are not ERC-20; they are internal claims tracked in the ‘shares‘ mapping.

5.3 Share Accounting

Each liquidity position is assigned a proportional share of the pool:

$$\texttt{liquidity} = \sqrt{R \cdot S} \quad \text{(initial)}$$

$$\texttt{liquidity} = \min\left(\frac{\Delta R \cdot \texttt{totalShares}}{R}, \frac{\Delta S \cdot \texttt{totalShares}}{S}\right) \quad \text{(subsequent)}$$

Positions can be topped up or partially withdrawn. Share balances are strictly bounded to 128 bits. No floating-point rounding is used.

5.4 Withdrawal

Withdrawals are fully proportional and performed via:

```
withdrawLiquidity(tokenA, tokenB, loc, liquidity, to, minA, minB)
```

The call burns up to the caller’s share of liquidity and transfers the underlying reserve and secured tokens to ‘to‘. Slippage guards may be specified for each token. Positions are deleted automatically if their share reaches zero.

No protocol fees are levied on entry or exit. LPs do not receive trade fee rebates; all fees are routed to protocol profit and FREE holders.

5.5 LP Non-Fungibility

PQSE does not tokenize LP positions as ERC-20s or NFTs. Each position is uniquely referenced by its integer ID. Ownership may be delegated to brokers, but cannot be fractionalized or bridged. This minimizes state complexity and avoids partial-share inconsistencies.

There is no LP transfer mechanism. LP positions are portable only via full withdrawal and re-deposit under a new identity.

5.6 Liquidity Role in Execution

During matching, the AMM pool is consulted only after limit orders and internal netting fail to clear all submitted inputs. The pool:

- Provides fallback execution within a price corridor.
- Ensures some liquidity is always available.
- Establishes a deterministic reference price for comparison.

The AMM does not compete with the book. It waits behind it.

6. Fee Accounting and Revenue Distribution

All trades in PQSE are subject to a deterministic fee model. The protocol splits each fee into three components: protocol revenue, liquidity rebate, and referral tip. Fee rates are fixed, quantized to 32-bit precision, and uniformly enforced.

6.1 Fee Structure

The total gross fee is charged on the reserve-side asset only. It is defined by:

$$\texttt{MAX\_FEE} = 6442450 \quad (\ 0.30\%)$$

Each trade applies this gross fee and splits it as follows:

- **Protocol:** 25% of the discounted fee.
- **Liquidity:** Remainder of the discounted fee.
- **Tip:** A separately computed referral fee with no discount.

Fees can be applied either:

- **Outside:** added on top of the amount transferred in.
- **Inside:** embedded in the traded amount (deducted from within).

Both modes are supported and are explicitly selected during execution.

6.2 Discount via FREE

Users may supply the `FREE` token during trade submission to receive a linear discount on the gross fee. The discount ramps from 0% to 100% over a domain of one full token unit:

$$\texttt{discountQ32} = \left\lfloor \frac{\texttt{FREE}_{\text{held}} \cdot 2^{32}}{10^{d}} \right\rfloor$$

where d is the token's decimal count.

Holding more than one unit yields no additional benefit. FREE balances are locked during the transaction.

6.3 Referral Tips

Any caller may specify a tip rate (capped at `MAX_TIP = 3%`) to be paid to their own address. This is intended to incentivize bundlers, interface providers, or transaction relayers. The tip is applied using the same inside/outside semantics as the base fee, but is never discounted.

All tips are paid immediately in reserve currency.

6.4 Revenue Distribution to FREE Holders

Protocol fee shares are accumulated in a 'profit' bucket per token. FREE token holders may claim their share at any time using:

`takeProfits(tokens[], speech)`

Each claim performs:

$$\texttt{share} = \left\lfloor \frac{\texttt{profit[token]} \cdot \texttt{FREE}_{\text{user}}}{\texttt{totalFREE}} \right\rfloor$$

The result is transferred immediately to the caller and deducted from the pool. Claims are idempotent, replayable, and non-sequenced. Any unclaimed dust remains with the protocol.

The required 'speech' field is a symbolic bytestring and must be non-empty. It may be used for compliance, logging, or signal embedding.

6.5 Determinism and Safety

Fee math is performed in Q.32 fixed-point format. All intermediate values use 256-bit arithmetic and are truncated to 64-bit outputs. Overflows revert. There are no percentage approximations or floating-point divisions.

6.6 Summary

- All fees are levied on the reserve token.
- Discounts are linear and capped at one unit of FREE.
- Protocol fees accrue to FREE holders.
- Tips are optional and immediate.
- LPs do not receive any portion of trade fees.

This fee model ensures bounded cost, forward-compatible rebate incentives, and closed-loop value routing for utility token holders.

7. Governance, Upgrades, and Immutability

PQSE is designed for execution determinism, not social flexibility. There is no upgrade path, administrative key, or mutable configuration. All protocol parameters are fixed in contract code and cannot be altered after deployment.

7.1 Governance Model

There is no on-chain governance.

- No DAO.
- No quorum.
- No vote.

The system implements no mechanism for stakeholders to change protocol behavior. This is intentional: markets require predictability, not opinion polls.

7.2 Upgrade Policy

There is no upgrade proxy or self-destruct path. The contract is deployed as a monolith with constructor-pinned references to its FREE token and zero external configurability. Future versions, if any, must be deployed as entirely separate systems.

Coexistence is possible; upgrades are not.

7.3 Parameter Fixation

The following values are hard-coded and cannot be changed:

- `MAX_FEE` — 0.30%
- `PRO_FEE` — 25% of discounted fee
- `MAX_TIP` — 3%
- `MIN_TIF` — 10 minutes
- `MIN_LIQ` — 1,000 units (both sides)

All numeric fields operate within strict bit-length bounds. There is no reliance on floating-point math or EVM precompiles. All matching logic is transparent and linearly verifiable.

7.4 Safety Disclosure

This design rejects upgradeability not out of dogma but necessity. In a post-quantum adversarial model, centralized upgrade paths are threat vectors, not features. Every mutable field is a surface; PQSE has none.

8. Conclusion

PQSE defines a minimal, deterministic framework for non-custodial securities exchange in adversarial environments. Its hybrid matching engine unifies limit orders and automated liquidity without resorting to off-chain sequencing, upgradeable control paths, or probabilistic routing.

All flows are transparent, all trades are 64-bit bounded, and all participants operate on equal footing. There is no governance, no treasury, and no discretionary control.

This is a brokerage protocol for systems that must survive asymmetric threats and asymmetric infrastructure.

It does not ask for trust.

It executes the best trade.