# QuantumSwap

## A Quantum-Resistant Decentralized Exchange on the QuantumCoin Platform

Revision 1, 2026

# Table of Contents

# 1. Abstract

QuantumSwap [2] is a decentralized exchange (DEX) [13] protocol built on the QuantumCoin blockchain [1][4], designed to operate securely in a post-quantum cryptographic environment [6]. It integrates NIST-standardized post-quantum cryptography at the protocol level, addressing long-term cryptographic risks posed by quantum computing.

QuantumSwap provides a modular, automated-market-maker-based trading ecosystem designed for both end users and token project creators, with tooling for decentralized token creation, liquidity provisioning, monitoring, and seamless exchange of tokens.

This document describes the design goals, cryptographic foundations, architecture, and functional components of the QuantumSwap protocol.

# 2. Introduction

Decentralized finance [5] has demonstrated the ability to remove intermediaries, reduce custodial risk, and provide global access to financial primitives. However, most existing DeFi protocols rely on cryptographic assumptions that may not remain secure in the presence of large-scale quantum computers [6][7]. Additionally, current DEX implementations face persistent issues such as front-running, sandwich attacks, and related forms of maximal extractable value [11], as well as fragmented developer tooling.

QuantumSwap is designed from the ground up to address these challenges. By building directly on the QuantumCoin blockchain, QuantumSwap leverages quantum-resistant cryptography, deterministic transaction ordering, and native protocol features to deliver a secure, extensible decentralized exchange intended to remain viable for decades.

The protocol is architected as a zero-trust, non-custodial, permissionless composition surface: privileged operations are eliminated at the protocol layer, settlement is consensus-verified, and every participant — whether a retail swapper, a liquidity provider, a token issuer, or an on-chain automated strategy — interacts with the same open, strongly-typed on-chain interfaces under the same rules, without requiring the cooperation of any off-chain operator, sequencer, or relay.

## 2.1 Design Goals

The rest of this document refers back to these goals:

- **G1. Quantum-resistant end-to-end.** All user-signed material uses NIST-standardized post-quantum signatures, and pool state is committed under NIST-standardized hash functions that are quantum resistant.
- **G2. Permissionless.** Any account may create a pool, provide liquidity, or execute a swap without allow-listing.

- **G3. Immutable core.** The AMM core (Factory, Pair, LP token) has no administrator, pauser, or upgrader.
- **G4. Deterministic ordering.** The order of execution within a block is fixed by QuantumCoin consensus rules, not by any sequencer, relay, or block builder.

---

## 3. Background: The Need for Quantum-Resistant DeFi

Classical DEXs authenticate swaps, liquidity-provider withdrawals, and governance votes with ECDSA or Schnorr signatures over elliptic-curve groups [12] such as secp256k1 or Ed25519. A sufficiently capable quantum adversary can recover a private key from an exposed public key via Shor's algorithm [7], enabling retroactive theft of long-lived liquidity positions and governance stakes. Because DEXs hold large pools of locked liquidity, manage high-frequency transaction flows, and depend on signature integrity and ordering, they are especially exposed to this risk.

QuantumSwap inherits the post-quantum cryptographic suite of QuantumCoin [1][16]:

- **Digital signatures.** ML-DSA (FIPS 204) [8] and SLH-DSA (FIPS 205) [9], chosen from the NIST Post-Quantum Cryptography standards [6]. Both have no known polynomial-time quantum attack.
- **Hash functions and extendable-output functions.** Keccak-based hash and extendable-output functions, as standardized in FIPS 202 [10], are used for address derivation and state commitments throughout the protocol.

By building on these primitives, QuantumSwap aims to ensure that:

- Transactions remain unforgeable in the presence of a quantum-capable adversary.
- Liquidity pools cannot be drained through forged signatures.
- Governance and protocol changes remain cryptographically verifiable.
- The DEX remains viable as long-lived public infrastructure.

The simultaneous use of ML-DSA and SLH-DSA, rather than either scheme in isolation, provides defense in depth across structurally distinct hardness assumptions: ML-DSA's security rests on the hardness of module-lattice problems, whereas SLH-DSA's security reduces to the collision and second-preimage resistance of its underlying hash function. A cryptanalytic advance against one family does not automatically compromise the other, so the composite signature surface diversifies cryptographic risk rather than concentrating it on a single algebraic assumption.

Post-quantum cryptography is an evolving field. QuantumSwap's cryptographic suite tracks the NIST PQC standards and is rotated only through consensus-enforced upgrades of the underlying QuantumCoin chain; the DEX itself has no cryptographic-agility switch at the application layer.

---

# 4. Platform Overview

## 4.1 QuantumCoin Blockchain

QuantumSwap is built on the QuantumCoin blockchain [1][4], a Layer 1 platform designed with post-quantum security as a core principle [16]. QuantumCoin integrates NIST-standardized post-quantum signature schemes (ML-DSA, SLH-DSA) and Keccak-based hash primitives into its base-layer protocol [16], and specifies its block production and transaction ordering rules in an accompanying consensus document [17].

The native coin of the platform, QuantumCoin, serves as the base settlement asset and the unit in which transaction fees are paid.

QuantumSwap inherits the security, ordering, and consensus guarantees of QuantumCoin.

## 4.2 Execution Model

QuantumSwap contracts execute inside the QuantumCoin virtual machine. Contracts are deployed at deterministic addresses derived from the deployer and a salt, and are invoked by transactions that carry the caller's ML-DSA and SLH-DSA signature, the target address, calldata, and fee parameters.

Execution is deterministic: every validator, given the same pre-state and transaction, produces the identical post-state. Contracts may synchronously invoke other contracts within a transaction, and all state changes produced by a transaction are applied atomically — on any failure, the entire call tree is reverted and no partial state is persisted. Fees for computation are paid in QuantumCoin by the transaction originator. Because state transitions are committed under consensus, the post-state of any successful transaction is identical across every validator in the network, and cross-contract composition is synchronous within a single atomic unit of work rather than mediated by asynchronous message passing.

## 4.3 Accounts and Addresses

An account address is derived from a Keccak-based hash of its post-quantum public key and is **32 bytes** wide, in contrast to the 20-byte addresses used by Ethereum-style chains [18]. The wider address space raises the collision and pre-image work factor well beyond what classical address schemes provide, and preserves meaningful margin against both classical birthday attacks and any future partial quantum speedups on the hash.

QuantumSwap inherits this address format directly from QuantumCoin: accounts, pair contracts, routers, and LP-token holders are all identified by the same 32-byte addresses used by every other contract on the platform, so no translation layer is required between QuantumSwap and the underlying chain.

---

# 5. QuantumSwap Architecture

QuantumSwap is a layered, non-custodial, zero-trust system composed of four loosely-coupled tiers. Each tier carries a single-responsibility mandate and communicates with its neighbors over strongly-

typed on-chain interfaces. No component in the Core or Periphery layers has any administrator, pauser, or upgrader.

The architecture follows a strict separation of concerns: pricing and settlement live in the Core layer, user-facing ergonomics and safety checks in the Periphery layer, auxiliary launch primitives in the Launch layer, and all human- and machine-facing interfaces in the Client layer. Contract bytecode is content-addressable and its deployment addresses are deterministically derived, so any caller can reconstruct the full dependency graph from the token addresses alone. The Core and Periphery layers expose only open, strongly-typed on-chain interfaces, which makes every cross-layer interaction statically checkable from the bytecode itself.

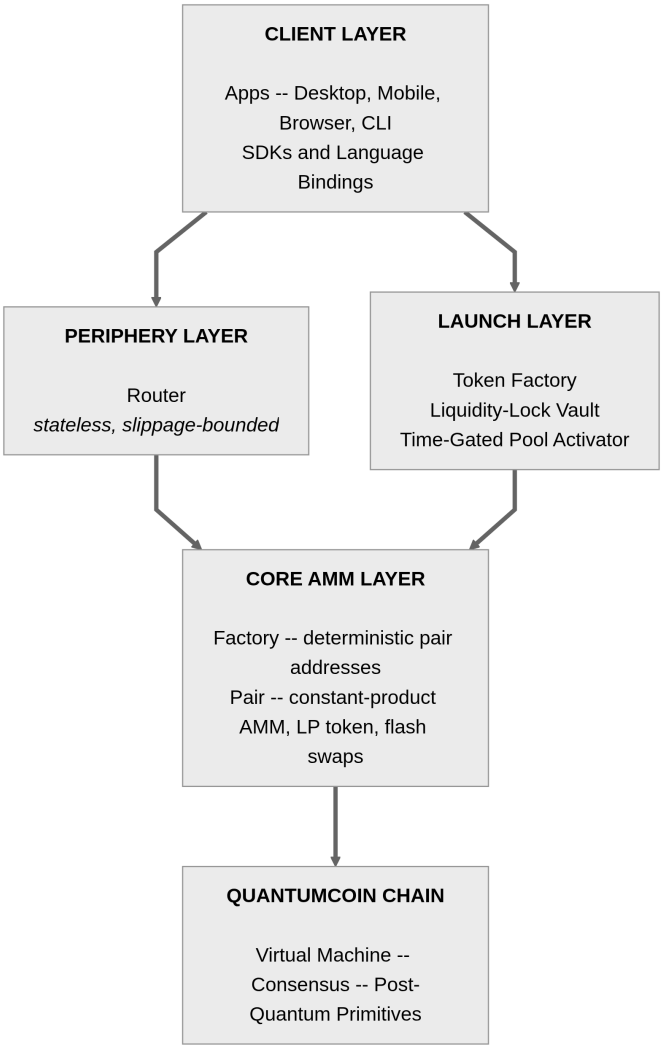


*Figure 1. QuantumSwap layered architecture: client, periphery/launch, core AMM, and the underlying QuantumCoin chain.*

1. **Core AMM Layer.** A singleton `Factory` contract deploys immutable `Pair` contracts at deterministic addresses derived from the canonically-ordered token tuple `(token0, token1)`, so that every unordered pair of tokens maps to exactly one canonical pair address. Each `Pair` is a two-token constant-product automated market maker that maintains `(reserve0, reserve1)` under the `x · y >= k` invariant, exposes reentrancy-guarded `mint`, `burn`, and `swap` entrypoints, emits indexed event logs on every state transition, and issues a fungible LP token

representing a pro-rata claim on the underlying reserves. The pair's initialization-code hash is fixed at factory deployment, so clients can derive any pair address off-chain from the two token addresses alone.

2. **Periphery Layer.** A stateless `Router` contract provides user-facing, composable entrypoints that sequence multi-hop swap paths over one or more pairs, enforce per-call slippage bounds (`amountOutMin` / `amountInMax`), and wrap or unwrap QuantumCoin on demand via a canonical wrapped-QuantumCoin pair. The Router holds no balances between transactions, emits fully-indexed event logs for every hop, and performs no implicit path discovery — every hop in a swap path is specified explicitly by the caller.
3. **Launch Layer.** A suite of standalone, immutable companion contracts implementing composable launch primitives: a fungible-token factory, a liquidity-lock vault providing time-release escrow of LP positions keyed to block height, and a time-gated pool activator. Each contract exposes only single-purpose state transitions with explicit pre- and post-conditions, is independent of the AMM core, and is independent of the others.
4. **Client Layer.** A set of isomorphic, strongly-typed SDKs and tooling — browser extensions, native desktop and mobile applications, CLI binaries, and language bindings — produced from a single canonical reference implementation via deterministic, reproducible builds. All client code is off-chain, open-source, and independently verifiable against the deployed on-chain bytecode.

---

# 6. Core Swap Functionality

## 6.1 Constant-Product Automated Market Maker

Each `Pair` contract maintains reserves `x` and `y` of its two tokens and operates as a constant-function automated market maker [3]. The pair enforces the constant-product invariant

```
x · y >= k
```

after every swap, evaluated on fee-adjusted balances, where k is the pre-swap reserve product. A swap fee `f` is deducted from the input amount before the invariant check, expressed as an integer numerator over a fixed denominator `FEE_DENOMINATOR` — i.e. the effective fee rate is `f / FEE_DENOMINATOR`. For an input amount `Δx` into reserve `x` producing output `Δy` from reserve `y`:

```
Δy = (y · Δx · (FEE_DENOMINATOR − f)) / (x · FEE_DENOMINATOR + Δx · (FEE_DENOMINATOR − f))
```

Both `f` and `FEE_DENOMINATOR` are defined as compile-time constants in the pair contract and are fixed at the time of deployment; neither is changeable post-deployment.

Reserves are stored as overflow-checked unsigned integers. Rounding is always *against* the trader and *in favor of* the pool, so the invariant is non-decreasing (setting aside the portion of the fee credited back to liquidity providers via the LP-token supply relationship).

The invariant is self-referential and path-independent: the post-swap state of the pool is a pure function of its pre-swap reserves and the swap's input amount, with no dependency on any external price signal, cross-contract quote, or historical state beyond the current reserves. Settlement is therefore fully local to the pair contract and is reproducible by any observer from the public on-chain state alone.

Liquidity is added by transferring both tokens to the pair and calling `mint`, which issues LP tokens proportional to the lesser of the two deposits relative to the existing reserves. Liquidity is removed by returning LP tokens and calling `burn`, which transfers the corresponding pro-rata share of both reserves back to the caller.

## 6.2 Token-to-Token and Token-to-QuantumCoin Swaps

QuantumSwap supports swaps between:

- two standard fungible tokens, via a direct pair or a multi-hop path; and
- a token and QuantumCoin, via a pair with a wrapped-QuantumCoin token that wraps and unwraps 1:1 inside the router.

The swap engine's execution is a function of pool state and transaction calldata alone.

## 6.3 Liquidity Locking

QuantumSwap provides liquidity locking to reduce rugpull risk. A liquidity provider transfers LP tokens to an immutable `LiquidityLockVault` contract together with an unlock block height. The vault has no administrator, no early-release path, and no privileged operator; the only state transition that returns LP tokens is a `release` call executed after the unlock block height has passed. Locks may optionally be assigned to a beneficiary address at creation time. The vault's complete specification is published alongside this whitepaper.

## 6.4 Time-Gated Swaps and Fair Launches

Pools may be configured to become active only at or after a specified **block height**. Time-based constraints are enforced by on-chain logic using block height rather than wall-clock timestamps, so they cannot be gamed by small timestamp manipulations by block producers. This enables fair launches, coordinated liquidity events, and reduced early-access manipulation for new token projects.

## 6.5 Decentralized Token Creation and Liquidity Bootstrapping

QuantumSwap includes built-in tooling for token project owners to:

- deploy a new standard-compliant fungible token from an immutable factory template;
- initialize a liquidity pool for that token; and
- release tokens and LP positions directly to addresses they control.

This process is fully decentralized and requires no custodial intermediary, off-chain operator, or allow-list.

## 6.6 Transaction Ordering and MEV

QuantumCoin orders transactions within a block using a deterministic rule fixed by consensus [17] (for example, by fee rate and then by transaction hash). This removes block-producer discretion in ordering and prevents arbitrary reordering attacks. Consensus-native ordering is a protocol-layer property, not a policy applied by a sequencer or relay, so it eliminates the entire class of attacks that depend on private mempools, discretionary block construction, or out-of-band transaction relays. It does **not**, by itself, prevent fee-bidding sandwich attacks — a form of maximal extractable value [11] — in which an attacker submits a higher-fee transaction before a victim's swap and a lower-fee transaction after it. Users are expected to protect themselves from sandwiching by setting slippage bounds ( `amountOutMin` / `amountInMax` ) on every swap via the Router (Section 6.8).

## 6.7 Flash Swaps

Every pair supports **flash swaps**. The pair optimistically transfers the requested output tokens to the caller, invokes a caller-supplied callback, and then verifies at the end of the transaction that either (a) the equivalent input amount has been paid in, or (b) enough of the output has been returned such that the constant-product invariant still holds after fees. Flash swaps are a decentralized primitive requiring no external operator; they enable atomic arbitrage, collateral swaps, and liquidation helpers built on top of QuantumSwap.

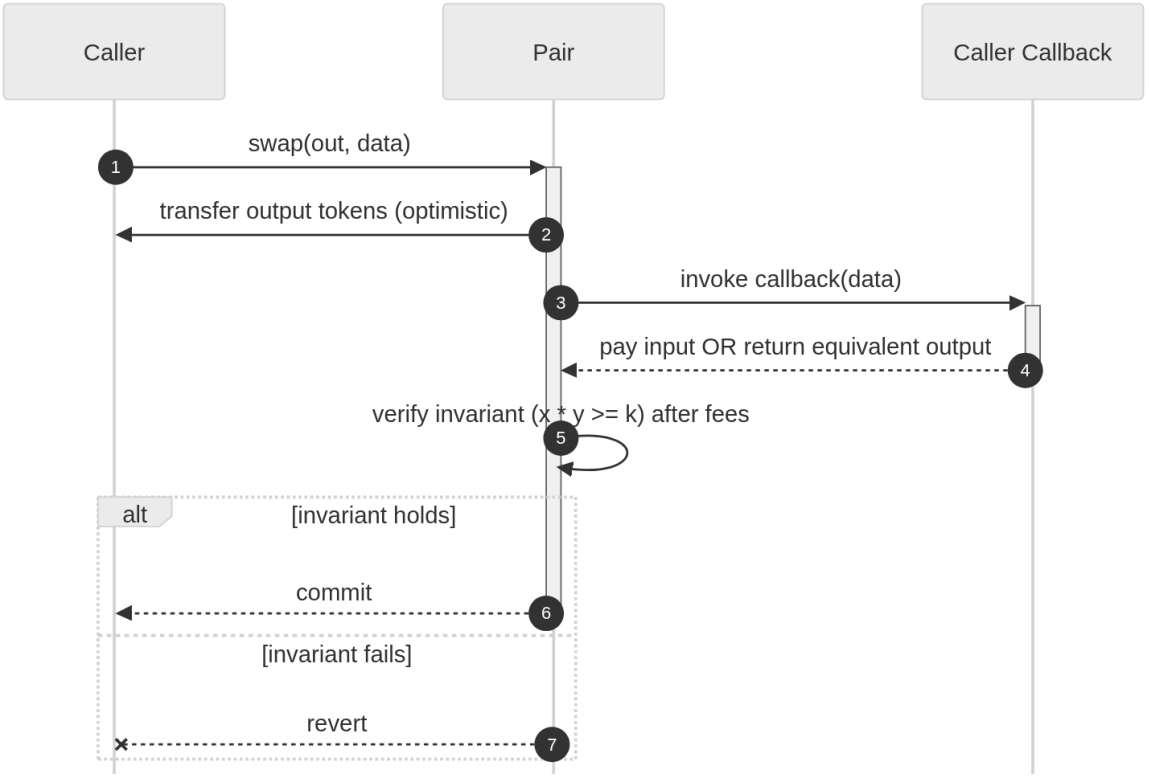


*Figure 2. Flash-swap execution sequence: tokens are transferred optimistically, the caller callback repays or returns, and the invariant is re-verified before the swap commits.*

### 6.8 Router Guarantees

The Router exposes the user-facing safety parameters as **mandatory** arguments on every swap call:

- `amountOutMin` / `amountInMax` — the worst acceptable price for the caller. The transaction reverts if the executed price is worse.
- `path` — the explicit ordered list of tokens to route through. The contract performs no implicit routing.

The Router is stateless between transactions and holds no user funds. It can be replaced by any alternative router without protocol cooperation.

### 6.9 Multi-Platform Access

QuantumSwap is accessible across multiple client environments:

- mobile applications;
- desktop applications;
- browser extensions;
- a command-line interface (CLI); and
- SDKs and APIs for third-party integrators.

All client code is open-source and reproducible. Clients communicate with the chain directly and do not rely on any QuantumSwap-operated backend.

---

## 7. QS Token Utility

The **QuantumSwap (QS) token** is the native utility token of the QuantumSwap ecosystem. It is a standard fungible token, issued on QuantumCoin, and designed to serve as a context-bearing reference identifier across the suite of QuantumSwap-adjacent applications, tooling, and third-party integrations. Possession of QS is permissionless.

These utilities are optional, evolving, and composable with — but not required by — the core protocol. The AMM core, the Router, and the Launch-layer contracts function identically regardless of whether any given account holds any QS; nothing in the core execution path reads, verifies, or branches on QS balances.

The token's role in the broader QuantumSwap ecosystem is that of a coordination primitive and context-bearing reference identifier: a shared namespace that downstream applications, analytics surfaces, and third-party integrators can anchor against without entangling the core AMM with any auxiliary accounting. Because the Core, Periphery, and Launch layers are entirely token-agnostic, any utility surfaced by QS is composable with — but strictly non-load-bearing for — the correctness, liveness, or settlement guarantees of the protocol itself, preserving the zero-trust property of the core execution path.

Detailed tokenomics — total supply, initial distribution, emission schedule, vesting, and related parameters — are published in a separate tokenomics document and are out of scope for this whitepaper.

QS is not intended to be, or to be the subject of, an investment contract, or security of any type.

---

## 8. Governance

QuantumSwap has no on-chain governance surface at the protocol layer. The core AMM (Factory, Pair, LP token) and the periphery Router are immutable once deployed: they have no administrator, no pauser, no upgrader, no allow-listing mechanism, no emergency-shutdown path, and no parameter-adjustment pathway. All protocol parameters — including the swap-fee rate — are committed at deployment and published alongside the deployed bytecode.

In the credible-neutrality sense, the absence of a governance surface is itself the governance property: because every protocol parameter is fixed at deployment and no component exposes an administrative, pausing, or upgrading entrypoint, there is no on-chain actor — QS holder, contributor, operator, or otherwise — whose cooperation is required for the protocol to operate, and correspondingly no on-chain actor whose compromise can alter the behavior of deployed QuantumSwap contracts.

Any broader coordination among QS holders, contributors, and third-party integrators is conducted off-chain and does not alter the behavior of deployed QuantumSwap contracts. Future versions of the protocol, if any, are deployed at new addresses and take effect only for users who voluntarily migrate to them.

---

## 9. Security Considerations

### 9.1 Threat Model

QuantumSwap's security posture is organized around defense in depth, least privilege, and a minimal trusted computing base: the set of components whose correctness is load-bearing for user safety is deliberately kept as small and as auditable as possible, and every additional layer is introduced only where it strictly reduces the attack surface rather than broadening it.

QuantumSwap's security model is designed against the following adversary capabilities:

- **Quantum-capable signature forgery** against pre-quantum signature schemes. Mitigated by QuantumCoin's ML-DSA and SLH-DSA signatures; no pre-quantum signature scheme is accepted anywhere in the protocol.
- **Block-producer reordering, censorship, or delay** of transactions within consensus rules. Partially mitigated by deterministic in-block ordering (Section 6.6).
- **Malicious or non-standard tokens** (reentrancy callbacks, fee-on-transfer, rebasing, transfer hooks). The pair contract assumes standard fungible-token behavior; non-standard tokens are **not**

**supported** in the core and must be wrapped before trading. LPs and traders are responsible for assessing token behavior before interacting with a pair.

### 9.2 Explicit Non-Mitigations

To avoid misleading users, the following are **not** defended against at the protocol layer:

- Fee-bidding sandwich attacks. Users must use Router slippage bounds to protect themselves.
- Impermanent loss. LPs bear price risk on both pool assets.
- Economic attacks exploiting non-standard token behaviors.
- Front-end or domain compromise. Users should verify contract addresses independently of any UI.

### 9.3 Engineering Practices

- Reentrancy guards on all external-call boundaries in the pair contract.
- Checked arithmetic; no unchecked math in the core.
- Deterministic, reproducible builds from a pinned toolchain. The on-chain bytecode hash is published alongside the source.
- Continuous AI-assisted contract auditing and static analysis of the pair, factory, router, and launch-layer contracts, run on every commit against a curated catalogue of known DeFi vulnerability classes (reentrancy, arithmetic overflow, rounding drift, access- control regression, unsafe external calls, and invariant violations) with findings triaged and published in the repository.
- A formal specification of the constant-product invariant, with machine-checked proofs where feasible.

---

## 10. Conclusion

QuantumSwap is a decentralized exchange designed for long-term viability in a post-quantum world. Its core is a constant-product automated market maker with an immutable, admin-less pair contract, a stateless router with explicit slippage guarantees, and flash-swap support. The protocol exposes no off-chain sequencing, and has no on-chain governance surface at the protocol layer: every parameter is committed at deployment. Its layered separation — Core AMM, stateless Periphery, immutable Launch primitives, and independently-verifiable Clients — keeps each tier's responsibilities narrow, its failure modes observable, and its integration surface minimal.

By combining post-quantum cryptography and a minimal, auditable, and fully on-chain AMM core, QuantumSwap aims to serve as foundational infrastructure for decentralized markets on the QuantumCoin blockchain.

QuantumSwap's advantage lies in its architecture, security model, and ability to enable decentralized exchange functionality in a future-proof manner.

---

## Appendix A. Glossary

- **AMM.** Automated Market Maker [14]. A smart contract [15] that quotes prices algorithmically from reserves rather than an order book.
- **Constant-product invariant.** The rule that the product of the two pool reserves does not decrease across a swap, up to fee adjustments.
- **Fair launch.** A token launch with no privileged pre-sale access, enforced on-chain.
- **Flash swap.** A swap in which output tokens are delivered optimistically and the corresponding input (or equivalent return) is verified at the end of the same transaction.
- **Front-running.** Inserting one's own transaction ahead of a pending public transaction in order to profit from its expected effect.
- **LP token.** Liquidity-provider token; a fungible receipt representing pro-rata ownership of a pair's reserves.
- **ML-DSA.** Module-Lattice-based Digital Signature Algorithm (FIPS 204).
- **Rugpull.** A malicious removal of pool liquidity by a token project's insiders, leaving traders with an illiquid token.
- **Sandwich attack.** A pair of transactions surrounding a victim swap, profiting from the price impact the victim creates.
- **SLH-DSA.** Stateless Hash-Based Digital Signature Algorithm (FIPS 205).
- **Slippage bound.** A trader-specified worst-acceptable output or input amount, enforced by the Router.

## Appendix B. References

Inline citations above refer to the numbered entries below.


1. QuantumCoin project. Project website. https://quantumcoin.org
2. QuantumSwap project. Project website. https://quantumswap.org
3. "Constant-function market maker." Wikipedia. https://en.wikipedia.org/wiki/Constant_function_market_maker
4. "Blockchain." Wikipedia. https://en.wikipedia.org/wiki/Blockchain
5. "Decentralized finance." Wikipedia. https://en.wikipedia.org/wiki/Decentralized_finance
6. "Post-quantum cryptography." Wikipedia. https://en.wikipedia.org/wiki/Post-quantum_cryptography
7. "Shor's algorithm." Wikipedia. https://en.wikipedia.org/wiki/Shor%27s_algorithm
8. National Institute of Standards and Technology. *FIPS 204: Module-Lattice-Based Digital Signature Standard.* 2024. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.pdf
9. National Institute of Standards and Technology. *FIPS 205: Stateless Hash-Based Digital Signature Standard.* 2024. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf
10. National Institute of Standards and Technology. *FIPS 202: SHA-3 Standard — Permutation-Based Hash and Extendable-Output Functions.*

2015. https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf

11. "Maximal extractable value." Wikipedia. https://en.wikipedia.org/wiki/Maximal_extractable_value
12. "Elliptic-curve cryptography." Wikipedia. https://en.wikipedia.org/wiki/Elliptic-curve_cryptography
13. "Decentralized exchange." Wikipedia. https://en.wikipedia.org/wiki/Decentralized_exchange
14. "Automated market maker." Wikipedia. https://en.wikipedia.org/wiki/Automated_market_maker
15. "Smart contract." Wikipedia. https://en.wikipedia.org/wiki/Smart_contract
16. QuantumCoin project. *Post-Quantum Security Whitepaper.* https://quantumcoin.org/whitepapers/post-quantum-security
17. QuantumCoin project. *Consensus Whitepaper.* https://quantumcoin.org/whitepapers/consensus
18. "Ethereum." Wikipedia. https://en.wikipedia.org/wiki/Ethereum
19. "UniSwap v2." Whitepaper https://app.uniswap.org/whitepaper.pdf