BLAZEPHOENIX PROTOCOL · LITEPAPER · FOR EVERYONE

BlazePhoenix

The swap aggregator and yield engine that does everything on-chain — no hidden servers, no administrator who can steal your funds, and a price quote that is guaranteed to match your actual swap.

SIMPLER THAN IT SOUNDS. MORE RADICAL THAN IT LOOKS.

Written by *Anonymous Mitra*

"The price you are shown is the price you get.
The funds in the vault are proven safe, on every transaction.
No-one — including us — can change that."

Version 1.0.0
June 2026
Ethereum · Base · Arbitrum · Optimism
License: BUSL-1.1
Anonymous Mitra

For the technical specification, see the BlazePhoenix Complete Technical Whitepaper.

What is BlazePhoenix?

Two things, built as one. A trading engine. A savings engine. Both running entirely on the blockchain — no company server needed, no hidden fees, no-one who can freeze your money.

THE TRADING ENGINE — THE AGGREGATOR

Best price. Every swap. Every chain.

When you want to swap one cryptocurrency for another, dozens of exchanges on the same network each hold a piece of the available liquidity. BlazePhoenix scans them all — automatically, on-chain, in the same transaction — and splits your trade across the best combination so you get the most output for your input. No manual searching. No trusting a company to do it honestly. The maths is in the contract, visible to anyone.

THE SAVINGS ENGINE — THE STAKING ENGINE

Lock. Earn. Borrow against it.

Lock your BZPX tokens for a period you choose and earn a share of the protocol's emission — a fixed pool of 180 million BZPX paid out over 7 years. You can also borrow against your locked tokens if you need liquidity, without selling. The contract's accounts are public, balanced, and enforced on every transaction. No trusted custodian. No dashboard you have to believe.

8	4	180M	0
EXCHANGE TYPES SUPPORTED	CHAINS, SAME CODE	BZPX REWARDS OVER 7 YEARS	OFF-CHAIN SERVERS NEEDED

The Problem with Crypto Today

Most "decentralised" apps are only pretending. Under the hood, important decisions are still made on private servers — and that means they can go wrong in ways you cannot verify.

Swapping: a solved problem that nobody solved right

> SIMPLE ANALOGY
>
> *Imagine you want to sell a car. There are 20 dealers in your city, each offering a different price. A good broker would visit all 20, find the best, and maybe even split the sale across two dealers to get you the maximum. The problem: most crypto "brokers" actually check prices on their own private database — which might be hours out of date — and show you the result. You have no way to verify they looked at all 20, or that they showed you the best one.*

MOST AGGREGATORS TODAY

Price is calculated on a company's server, then sent to the blockchain to execute. The price shown and the price you actually get can differ — and there's no way to prove they shouldn't have. If the server goes offline, you can't swap.

BLAZEPHOENIX

Price is calculated *inside the blockchain transaction itself* — the same computation that executes the trade also determines the price. They cannot differ, because they are the same operation. Nothing happens off-chain.

Staking: where trust is most abused

TYPICAL STAKING PROTOCOL

You lock tokens with a smart contract. But an admin key can upgrade the contract, change the rules, or — in the worst cases — drain it. The protocol's solvency is shown on a dashboard you have to trust. Liquidations depend on bots someone must keep running.

BLAZEPHOENIX STAKING

No admin can touch your principal — the function that would do it doesn't exist in the code. Solvency is a number on the blockchain, checked automatically on every transaction. Liquidations happen without bots, paid for by ordinary user activity.

THE REAL COST OF TRUSTING SERVERS

Over $500 million drained in the last three years from DeFi protocols through the exact failure modes BlazePhoenix removes structurally: oracle manipulation, upgradeable-proxy exploits, and keeper failures. These are not edge cases — they are the predictable cost of off-chain architecture, paid by users, repeatedly, because the root cause was never fixed.

How the Swap Works

Here is what actually happens when you swap two tokens through BlazePhoenix — from the moment you click to the moment the tokens arrive.

Step 1 — Finding the pools

Every exchange on Ethereum and its Layer-2 networks is a "pool" — a smart contract holding two tokens in a fixed mathematical relationship. Pools are discovered automatically: the first time a pool is used through BlazePhoenix, it is added to a self-updating registry. The registry never needs a human to maintain it. Pools that go unused decay out and are replaced by fresher ones. All of this happens inside the blockchain, not on a server.

Step 2 — Finding the best price

Not all pools holding the same two tokens quote the same price. Before executing, the Solver (a piece of read-only code that touches no funds) evaluates every registered pool for your pair and finds the combination that gives you the most output. It does this entirely from public blockchain data — no external feed, no off-chain optimiser.

> HOW THE PRICE STAYS HONEST
>
> *For simple pool types (constant-product, Solidly, stable-swap), the price is calculated by a direct formula — like solving an equation. For concentrated liquidity pools (Uniswap V3, V4, Camelot), the price is extracted by actually running the pool's own swap code in a test mode and reading the result before any money moves. This is called the Exact Pass. The quote cannot differ from the fill because the quote IS a test execution of the fill.*

Step 3 — Splitting across pools

If one pool is not deep enough to fill your whole order without moving the price significantly, BlazePhoenix splits your trade across multiple pools in the same transaction — up to five at once, or two hops through a bridge token like WETH or USDC. The split is sized proportionally to each pool's real depth, so shallow pools get less and deep pools get more.

Step 4 — Executing atomically

The Router — the only contract that ever holds your tokens, and only for milliseconds — executes the entire route as a single atomic transaction. Either the whole thing succeeds and you receive at least the minimum output you approved, or the whole thing reverts and nothing changes. There is no partial fill, no stuck state, no way for the protocol to take your tokens without delivering.

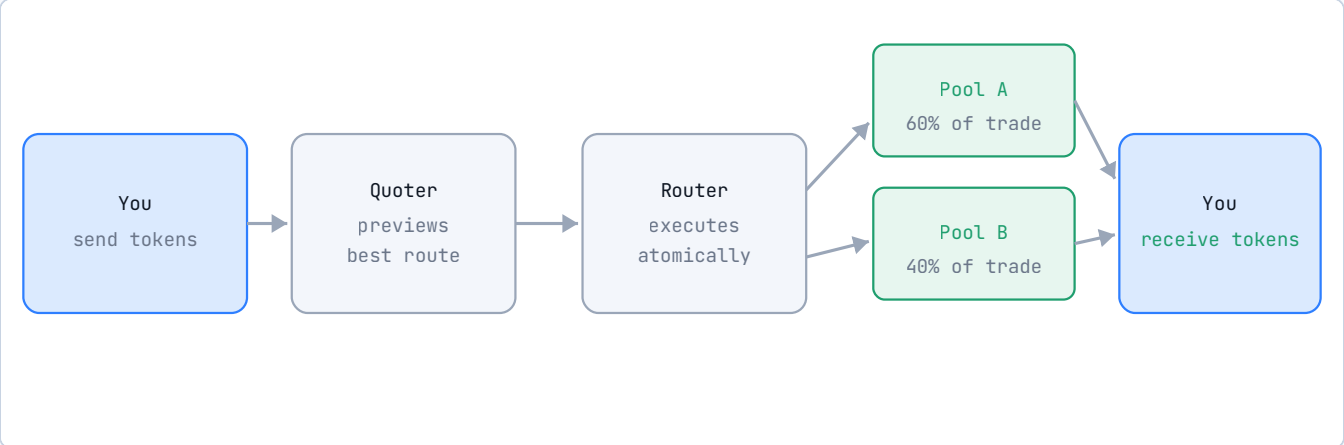

FIG. 1 *A split swap through two pools in one atomic transaction. The Quoter previews the best route; the Router executes it; either you receive at least your minimum or the transaction reverts entirely.*

Why This is Different From Everything Else

Seven things that most aggregators get wrong — and what BlazePhoenix does instead. No jargon. Just the mechanism and why it matters to you.

1 **The price shown is the price you get.** Other aggregators quote prices from a server, then try to match them on-chain. Delays, slippage, MEV bots — all create gaps. Here the quote is produced by running the actual swap in a test and reading the result. There is no gap to exploit.

2 **No database of pools to go stale.** Most protocols maintain a list of pools on a server that needs constant updates. If the server lags, you trade in a stale market. This protocol learns about pools by trading through them. The list is always current because trading keeps it current.

3 **Cannot be censored.** An RFQ server can silently refuse to quote a specific wallet, token or country. There is no RFQ server here. The quoting code is a public function on the blockchain — it cannot refuse. If the chain runs, the quote runs.

4 **Cannot be upgraded to steal from you.** Many contracts use an "upgradeable proxy" — the admin can replace the code in one transaction, changing the rules after you've deposited. This contract has no proxy and no upgrade path. The code you audit today is the code that runs forever. An optional one-way switch can permanently seal even the admin powers.

5 **Front-running is much harder.** MEV bots exploit the window between an off-chain route announcement and the on-chain execution. Here, the route is computed on-chain at execution time — there is no pre-published path for a bot to front-run.

6 **Fake pools cannot manipulate the price reference.** Attackers deploy dozens of empty pools quoting a false price, hoping to make them the "median" and mislead the router. This protocol uses the pool with the *largest real balance* as the reference — faking that requires actually depositing real money, at which point arbitrage fixes the price anyway.

7 **Exactly the same on every chain.** The identical code runs on Ethereum, Base, Arbitrum and Optimism. One audit covers all of them. A bug found anywhere is fixed everywhere. You get the same guarantees regardless of which chain you use.

THE BIGGER PICTURE

None of these seven properties can be added later. They are structural — you either design the protocol without off-chain dependencies, or you don't. Every aggregator that uses an off-chain solver, a keeper, or an upgradeable proxy has traded these guarantees for convenience, and has accepted the failure modes that come with it. BlazePhoenix made the opposite choice from the start. You cannot retrofit trustlessness; you have to build it in.

The Staking Engine — in Plain Language

You lock BZPX tokens. You earn yield. You can borrow against your stake if you need cash. The contract keeps its own accounts honest — automatically, every transaction, forever.

The lock and the boost

Unlike most staking protocols, BlazePhoenix has **no flexible staking**. When you deposit, you choose a lock period: anywhere from 90 days to 7 years. The longer you commit, the bigger your share of the rewards — not as a fixed tier, but on a smooth curve. Someone who locks for 5 years earns double the rewards per token compared to someone who locks the minimum. Someone who locks the maximum (7 years) earns 2.75× as much.

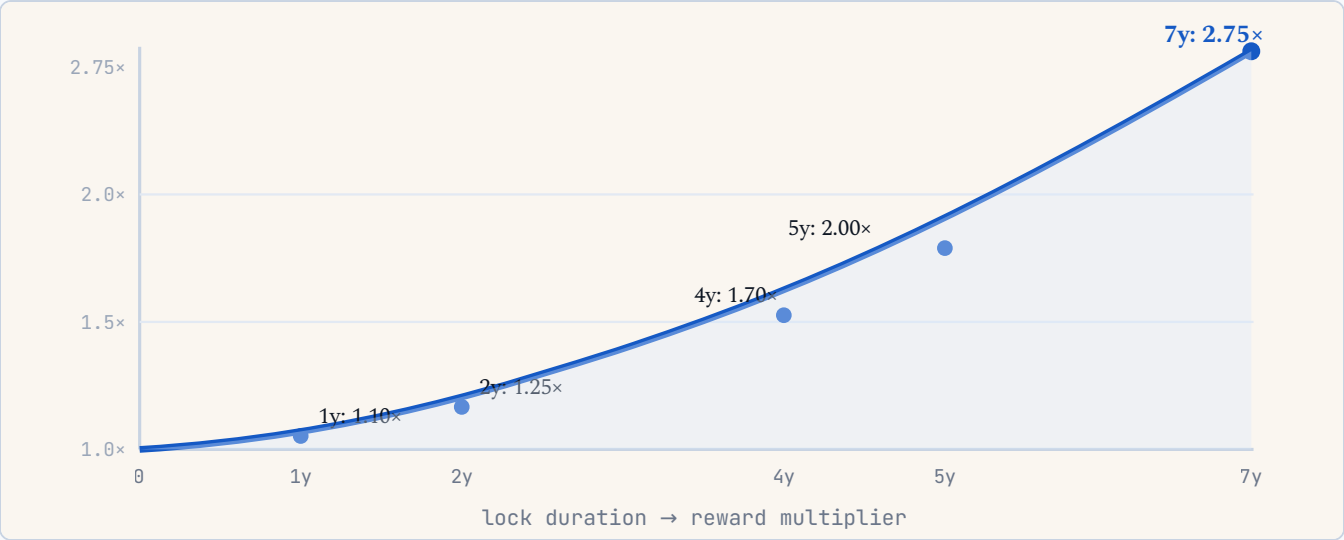

FIG. 2 *The reward multiplier as a function of lock duration. Longer commitment earns a proportionally larger share of the emission pool. The curve is smooth — no arbitrary "tiers." Commit 5 years and earn 2× per token versus the minimum lock.*

Two sources of yield

Stakers receive rewards from *two entirely separate sources* that never mix:

SOURCE 1 — EMISSION

Protocol rewards

A fixed pool of 180 million BZPX released linearly over 7 years. Your share depends on your locked amount and your boost multiplier. Every staker gets this, whether they borrow or not.

SOURCE 2 — BORROWER INTEREST

Interest income

Stakers who *don't* borrow also earn the interest paid by stakers who *do* borrow. Think of it like being the lender: you supply the collateral base, and the borrowers pay you for using the liquidity you didn't need.

The two streams are calculated by two entirely separate formulas — *accRewardPerShare* for emission and *accPureYieldPerShare* for interest — and they never mix. There is no hidden cross-subsidy: every unit of interest a pure staker receives is a unit that a borrower paid, and every unit of emission a borrower receives is funded by the fixed protocol budget that was pre-allocated at genesis. You can see which stream is paying you at any moment by calling the contract's view functions directly on-chain, with no login required.

Borrowing, Safety and the No-Backdoor Guarantee

Borrowing against your stake

If you need liquidity but don't want to sell your BZPX, you can borrow up to **50%** of your locked stake value. The interest rate rises with demand — low when few people borrow (about 1%), steep when the protocol is nearly fully borrowed (up to 150% at the extreme). This protects everyone: high rates when liquidity is scarce encourage repayment, keeping a withdrawal buffer for stakers who want out.

> NO PRICE ORACLE NEEDED — THE UNUSUAL PART
>
> *Most lending protocols need to know the current market price of your collateral to decide if you're undercollateralised. That price typically comes from an "oracle" — an external feed that has been manipulated to drain over $320 million from DeFi protocols since 2022. BlazePhoenix avoids this entirely: you borrow the same token you staked (BZPX). So there is no price to look up. Your loan-to-value is just: how much BZPX do I owe vs how much BZPX is locked. Simple arithmetic. No oracle. One entire attack surface, gone.*

What happens if you don't repay

Interest is charged continuously and deducted from your staked balance (the position "self-amortises"). If your debt grows to 95% of your stake, anyone can liquidate your position: your debt is repaid from your stake, the liquidator receives a 5% bonus for doing so, and any remaining stake is returned to you. The contract keeps a running estimate — visible to anyone — of how many days you have before liquidation at the current rate.

The three safety guarantees

✓ **Provably solvent, every transaction.** Before and after every deposit, withdrawal, borrow, repay or claim, the contract checks that its actual token balance equals the sum of everything it owes. If it doesn't, the

transaction reverts. Insolvency is not "monitored" — it is impossible to reach through ordinary use.

✓ **No admin can take your principal.** There is no function in the contract that lets anyone — including the team — move a staker's locked tokens to an arbitrary address. The only way funds leave your account is through your own action (withdrawal after lock expiry) or liquidation (only if your LTV exceeds 95%). This isn't a promise. The function literally does not exist.

✓ **No bots needed to stay safe.** Most lending protocols depend on keeper bots to scan for undercollateralised positions and liquidate them. If the bots fail, bad debt accumulates silently. Here, every ordinary user transaction (any deposit, claim, borrow) performs a small automatic sweep of borrower positions, liquidating any that qualify. The book maintains itself.

Emergency exit

If a critical vulnerability is ever found, an emergency mode can be activated that opens a simple withdrawal function: read your balance, send it to you, zero the record. This path is deliberately simple — it does not touch the complex reward and interest calculations, so even if *those* contained a bug, the withdrawal works. There is a mandatory 30-day window before any admin can access remaining funds, so every staker can exit first. The admin's "recovery" function reaches only what stakers chose to leave behind.

Three Words That Mean Something

*The protocol is described in three words: **stateless, serverless, ungovernable.** Here is what each actually means — not as marketing, but as verifiable properties of the code.*

STATELESS

The maths is a pure function.

The pricing library — the Core — has no storage, no owner, no upgrade mechanism. It is a collection of mathematical functions. The same inputs always produce the same outputs. There is no state to corrupt, no admin to retune a constant, no hidden variable that changes the answer. When you audit the Core, you have audited it permanently — there is nothing that can change afterwards.

SERVERLESS

No process that needs to keep running.

Discovery, routing, pricing, settlement, liquidation maintenance — all happen inside blockchain transactions. There is no server to go offline, no database to fall stale, no bot that must be kept running. The protocol works as long as the blockchain works. The team going on holiday changes nothing.

UNGOVERNABLE

Governance only where it is harmless.

The team can add new exchanges and expand the venue list. That is safe: more choices is always better for users. But the team *cannot* change what the contract charges, redirect the protocol fee to a different address, pause your ability to

withdraw, or modify the maths. And through a permanent, irreversible switch in the contract, even those grow-only powers can be surrendered forever. Once walked through, that door cannot be re-opened.

TEST IT YOURSELF

Each of these three properties is falsifiable. You can inspect the Core library on Etherscan and confirm it has no storage variables and no owner. You can run the Quoter's view function and confirm it makes no external calls. You can look for the `renounceControl` function and call it on a testnet to confirm it is irreversible. These are not marketing claims — they are properties of deployed bytecode that anyone can read, call and verify. Stateless. Serverless. Ungovernable.

BZPX — The Token

One billion BZPX. Fixed forever. The largest share goes directly to market liquidity — not to the team, not to investors. Here is the full picture.

Total supply: 1,000,000,000 BZPX. No more, ever.

There is no inflation mechanism. The staking rewards come from a pre-allocated slice of the fixed supply, not from newly minted tokens. Once the 180 million emission slice is paid out over 7 years, no more BZPX can be created — not by the team, not by governance, not by any mechanism. The scarcity is in the contract, not in a policy.

FIG. 3 *BZPX allocation. 60% goes straight to market liquidity — the float that lets the aggregator route through it. The team's 11.3% is the only insider tranche that vests over time (30 months). Everything else is liquid at launch.*

WHAT MOST PROTOCOLS DO

Team gets 20–40% with a short lock-up. Large "ecosystem fund" controlled by the team. Emission schedule that can be changed by governance. Inflation that dilutes holders indefinitely.

BZPX

Team gets 11.3%, vested over 30 months. Emission is from a fixed pre-allocated slice — no new minting. No governance mechanism to change any of this. Supply is a property of the contract.

Because the staking emission is carved from the fixed supply and the team tranche vests linearly, the fully diluted supply is known and fixed from genesis — there is no mechanism, administrative or otherwise, to mint beyond one billion BZPX. The token's scarcity is a property of the contract, not a policy that can be revised. 60% going to market liquidity is the most unusual feature of the allocation — it reflects the protocol's conviction that the aggregator needs a liquid, deep market to route through, and that inflating the team's allocation at the expense of that depth would be self-defeating.

Trust Nothing — Verify Everything

You don't have to take our word for any of this. Here is exactly what you can check yourself, for free, in seconds, using public blockchain tools.

Things you can verify right now

→ **Is the staking contract solvent?** Call `isSolvent()` on the staking contract. Returns `true` or `false`. No login required. No dashboard to trust. One blockchain read.

→ **How many days until my position is liquidated?** Call `daysToLiquidation(yourAddress)`. Returns an exact number of days based on your current debt, stake, and the prevailing interest rate. Updated every block.

→ **What are my pending rewards?** Call `pendingRewards(yourAddress)` and `pendingPureYield(yourAddress)`. See exactly what you've earned before claiming. No estimate — exact, from the contract's accumulators.

→ **Does the admin have dangerous powers?** The contract's source code is verified on public explorers (Etherscan, Basescan). Search for functions that could redirect staker funds. You won't find them. The absence is the proof.

→ **Is the price I'm being quoted fair?** The Quoter contract is open source. You can run the same function it runs — it makes no off-chain calls. The number you see in the UI is produced by the same code that will execute your trade.

The audit bitmask

For the technically curious: calling `auditInvariants()` on the staking contract returns a number from 0 to 31. A return of `0` means five separate internal consistency checks all pass simultaneously: the contract is solvent, the reward budget hasn't been over-spent, the accumulators only ever increase, and the reward share calculations are bounded correctly. Any other number means at least one check has failed — which would trigger the permissionless circuit-breaker mechanism and allow any user to halt the contract.

> THE PRINCIPLE
>
> A protocol that custodies your money should not ask to be trusted. It should publish the means to verify it. BlazePhoenix makes its entire integrity checkable with public blockchain calls that require no account, no API key, and no trust in any intermediary. The blockchain is the source of truth — not our website, not our Twitter, not our word.

Common Questions

Is BlazePhoenix a company? Who is behind it?

The protocol is published under the name *Anonymous Mitra*. The code is open, the contracts are verified on public explorers, and the economics are enforced by the contracts themselves — not by the goodwill of any named individual or company. This is a deliberate design choice: a protocol whose safety relies on its operators' identity is weaker than one whose safety relies on its mathematics.

Can the protocol be shut down?

No external party can shut down the swap aggregator — there is nothing to shut down. The contracts are deployed and immutable. As long as Ethereum (or Base, Arbitrum, Optimism) runs, the contracts run. The staking contract has a pause mechanism for emergencies, but even while paused, stakers can always withdraw their principal through the emergency exit.

What happens if a bug is found?

The staking contract carries a pull-only emergency hatch: stakers can always withdraw their principal through a simple, deliberately minimal code path that bypasses the complex reward logic. The admin can declare an emergency, which opens this path. A 30-day window must pass before any admin could access remaining funds — meaning every staker exits first.

How does the 0.28% fee work?

The fee is charged on the quoted output — the amount the protocol told you you'd receive. If you actually receive *more* than the quote (because of favourable rounding or price movement), the surplus is yours, entirely fee-free. The protocol never profits from the difference between the price it promised and the price it delivered.

Is this the same as Uniswap, 1inch or Paraswap?

It aggregates across Uniswap (and many others), like 1inch and Paraswap do. The difference is architectural: those protocols use off-chain systems to do the hard work. This protocol does everything on-chain. The trade-off: slightly more gas, in exchange for the certainty that what you see is what you get, and no server can go down, censor you, or be compromised between your quote and your fill.

Can I use it without reading any of this?

Yes. The interface works like any other aggregator — input token, output token, amount, swap. The difference is invisible in normal usage and only matters when something goes wrong: unlike other aggregators, when something goes wrong here, it goes wrong in a way you can verify on-chain, and in a way the contract was designed to prevent. You don't need to understand the mechanism to benefit from the guarantees it provides.

Quick Reference Glossary

Terms you might encounter — in plain language.

Aggregator
A tool that checks multiple exchanges at once and sends your trade to the best combination — all in one transaction.

AMM
Automated Market Maker. An exchange that runs as a smart contract with a formula. No order book, no humans. Price is determined by the ratio of tokens in the pool.

Atomic
Either the whole transaction succeeds, or nothing changes. No partial fills. You cannot lose tokens without receiving the output.

Boost multiplier
A multiplier on staking rewards based on lock duration. 2× means you earn twice as many rewards per token as the minimum locker.

Emission
The fixed pool of tokens released gradually to stakers as rewards. BlazePhoenix emits 180M BZPX over 7 years. Once exhausted, no more is created.

Iron-Law floor
A minimum acceptable output. If the best route would return too little, the protocol refuses to execute rather than complete a bad trade.

LTV (Loan-to-value)
The ratio of how much you borrowed to how much collateral you have. At 50%: your loan = half your stake. At 95%: the contract liquidates you.

Oracle
An external price feed that tells a smart contract what something is worth. BlazePhoenix staking needs no oracle because collateral and debt are the same token.

Permissionless
Anyone can do it. No approval, no whitelist, no account required. The circuit-breaker can be tripped by any wallet in the world if the contract is provably insolvent.

Proxy / upgradeable
A contract whose logic can be replaced after deployment. Useful for bugs, but creates risk: the admin can change the rules after you deposit. BlazePhoenix has no proxy.

Revert
A failed transaction that cancels all its effects. If a swap reverts: nothing happened, no money moved, no state changed.

Slippage

The difference between the price when you request a swap and when it actually executes. Large trades move the pool price; BlazePhoenix accounts for this precisely.

Vitality

The score the protocol gives each pool based on activity and depth. High-vitality pools are preferred in routing; idle pools decay in rank and get replaced.

Stateless

Code with no memory between calls — it only produces output from its inputs. The Core pricing library is stateless: it stores nothing and cannot be configured by anyone.

This litepaper summarises the BlazePhoenix protocol for a general audience. For the complete mathematical specification, see the Technical Whitepaper.

Anonymous Mitra · June 2026 · Version 1.0.0 · BUSL-1.1