TokSol

The TokSol Protocol

Bidirectional cryptographic tokens as a settlement layer for real-world asset tokenization

VERSION 1.0 · JULY 2026

ABSTRACT

TokSol is a tokenization-engineering house. We take a proven on-chain bidirectional-token engine — bonding curves implemented as Anchor programs on Solana — and apply it, asset by asset, to the tokenization of real-world assets. For each client we study the underlying asset and engineer a bespoke platform, a bespoke protocol, and a custom bonding curve calibrated to it. This document describes the mechanism: how a bonding curve prices a token against a base asset as a deterministic function of circulating supply, why that construction makes liquidity permanent and two-way, and why the reserve backing every token is a mathematical property of the system rather than a contractual promise. It is a technical description of an engine and its guarantees, not a description of any single deployment.

1. Abstract

The tokenization of real-world assets has moved from thesis to infrastructure, yet most tokenized assets share a single fatal flaw: they can be issued but they cannot be exited. TokSol addresses this by applying a bidirectional cryptographic token — a bonding curve that prices both purchases and redemptions against a base asset — as the market layer for tokenized real-world assets. Because the same continuous curve governs entry and exit, every holder can always transact against the protocol itself, with no dependence on finding a counterparty and no dependence on an external liquidity provider.

This paper sets out the definitions, the mathematics, and the on-chain architecture of the engine. The central result is a solvency property: at every moment the reserve holds exactly the funds required for every outstanding token to be redeemed, because the reserve is by construction the integral of the price function over the circulating supply. We then describe how TokSol engineers a bespoke platform, protocol, and custom curve for each client asset, and we are explicit about where the guarantee ends — it covers the token and its reserve on-chain, not the enforceability of the off-chain asset.

2. Introduction

Real-world asset tokenization is the representation of ownership or economic exposure to an off-chain asset — real estate, credit, commodities, equipment, receivables, royalties — as a programmable token on a public blockchain. The appeal is structural: fractional ownership, programmable transfer, continuous settlement, and an auditable record of holdings replace paper title, bilateral negotiation, and multi-day clearing. Institutional analysis has been directionally consistent on the trajectory. Work from BCG, Standard Chartered, and McKinsey projects the on-chain tokenized-asset market growing from the low tens of billions of dollars today toward the trillions within the decade. These are dispersed projections across differing definitions and assumptions, not a single figure and not a guarantee; but the direction of the estimates is unusually consistent across independent institutions.

The obstacle is not issuance. Minting a token that represents a building or a loan is straightforward, and the industry has demonstrated it thousands of times. The obstacle is liquidity. A tokenized asset that cannot be sold is not an asset in any economic sense — it is a certificate. Most tokenized assets die of illiquidity: they are issued into an order book that never forms, or a pool that is never funded, and the holder discovers that programmable ownership without a market is simply a more elaborate way of being stuck. Traditional market structures assume a counterparty on the other side of every trade, and for long-tail, high-friction, or newly issued assets that counterparty frequently does not exist.

TokSol's position is that the liquidity problem must be solved at the protocol layer, not left to market formation. If the mechanism that issues a token also stands ready to redeem it — deterministically, on-chain, at a price it computes itself — then liquidity is a property of the instrument rather than a hope about its market. The rest of this document describes how that is done and what it does and does not guarantee.

3. Bidirectional cryptographic tokens

A bidirectional cryptographic token is a token whose price is defined by a bonding curve. A bonding curve is an automated market maker (AMM) that prices a token against a base asset — in our deployments, a stablecoin such as USDC — purely as a function of the token's circulating supply. When a buyer sends base asset to the curve, the protocol mints new tokens and the spot price rises along the curve; when a holder sells, the protocol burns their tokens, returns base asset from the reserve, and the spot price falls along the same curve. There is no order book, no matching engine, and no discretionary market maker. The curve is the market.

Because the curve prices both sides of every trade, liquidity is permanent and two-way. A holder never has to find a buyer: the protocol itself is always the counterparty, and it will always execute a redemption at the current curve price so long as the holder is burning tokens. This is the decisive difference from conventional AMMs that depend on liquidity providers depositing inventory — providers who can withdraw, whose pools can drain, and whose absence leaves a token untradeable. Under a bonding curve there is no external liquidity provider to disappear; the

reserve that funds redemptions is accumulated from purchases and is bound to the token by the mathematics of the curve.

Two-way liquidity matters specifically for real-world assets because RWAs are the assets least likely to develop a natural secondary market. A single tokenized warehouse, a niche receivables pool, or a fractionalized equipment fleet may have very few natural buyers at any given moment. In a counterparty-dependent venue this thinness is fatal. Under a bonding curve, thinness of the natural market is irrelevant to the holder's ability to exit: the exit path runs through the protocol and the reserve, not through another human being. The holder always exits at the curve price, computed on-chain, without any external liquidity provider standing between them and their funds.

Circulating supply (S)

The total number of tokens currently minted and outstanding. It rises when the curve mints tokens on a buy and falls when it burns tokens on a sell. Every price on the curve is a function of S.

Reserve (R)

The pool of base asset held by the protocol to fund redemptions. It grows as buyers deposit base asset and shrinks as sellers withdraw it. Its size is determined entirely by the curve and the current supply.

Base asset

The unit of account and medium of settlement against which the token is priced — a stablecoin such as USDC in TokSol deployments. Buys are paid in the base asset and redemptions are returned in it.

Spot price (P)

The instantaneous price of one token in units of the base asset, given by the price function evaluated at the current circulating supply. It is the marginal price of the next infinitesimal unit, not the average price of a finite trade.

Invariant

The fixed mathematical relationship between reserve and supply that the curve preserves through every transaction. Every buy and every sell is priced so that this relationship still holds afterward, which is what keeps the reserve exactly sufficient to redeem all outstanding tokens.

4. Mathematical foundations

The behaviour of a bidirectional token is fully specified by two functions: a price function that maps circulating supply to spot price, and a reserve function that maps circulating supply to the base asset held against it. TokSol's reference engine uses a square-root price function. The exponent is a design parameter chosen per asset; the reference form is presented here.

```
P = S^0.5
```

Price function. The spot price of the next token is the square root of the circulating supply. Price rises with supply, but with a decreasing marginal rate.

The reserve required to support a given supply is the area under the price function from zero to S — the total base asset that must have been paid in to move the supply from nothing to its current level. Integrating the price function gives the reserve function.

```
R = ∫₀^S x^0.5 dx = (2/3) · S^1.5 ≈ 0.666667 · S^1.5
```

Reserve function. The reserve is the definite integral of the price function over the circulating supply. This is the exact base-asset balance the protocol must hold to redeem all S tokens.

Every transaction is priced so that after it executes the pair (S, R) still satisfies the reserve function — this is the invariant. Given a purchase that adds ΔR of base asset to the reserve, the new supply is found by inverting the invariant; given a redemption that moves supply by ΔS, the resulting reserve follows directly. The two relations below are equivalent expressions of the same invariant, one solved for the new supply after a reserve change and one solved for the new reserve after a supply change.

```
S_new = ( S^1.5 · (R + ΔR) / R )^(2/3)
R_new = R · (S + ΔS)^1.5 / S^1.5
```

Invariant pricing. A buy of ΔR base asset determines the new supply; a supply move of ΔS determines the new reserve. Both are exact and deterministic.

Several properties follow directly from this construction. Early buyers transact at a low price, because near S = 0 the price function is small; the price is not set by negotiation but read off the curve. As supply grows the increases decelerate — the marginal price rises with the square root of supply, so each additional unit of supply moves the price less than the last. The curve is continuous and deterministic: for any state and any trade size there is exactly one output, computable on-chain, with no dependence on external quotes. And critically, the invariant holds regardless of the order or the size of transactions. A large buy followed by a small sell, or a thousand small trades in any sequence, all leave the reserve exactly consistent with the resulting supply, because each step is priced against the same continuous curve.

The Solvency Theorem

At all times, the reserve holds exactly the funds required for every holder to sell simultaneously. Solvency is a mathematical property, not a contractual promise.

The reason the theorem holds is structural rather than procedural. The reserve is, by construction, the definite integral of the price function over the circulating supply — that integral is not an accounting estimate of what should be on hand, it is the exact amount that was paid in to reach the current supply. Redeeming all S tokens means integrating the same price function back down from S to zero, along the same curve, which returns exactly R. There is no path through the curve that leaves the reserve short of the amount needed to burn every outstanding token, because the amount needed is defined as that integral and the reserve is maintained equal to it at every step. Solvency is therefore not something the protocol promises to honour; it is a quantity the protocol cannot make false without violating its own arithmetic.

5. Protocol architecture

The engine is implemented as a set of on-chain programs written in Rust using the Anchor framework and deployed to Solana. Solana was chosen for engineering reasons specific to the workload. Real-world asset markets generate high-frequency, low-value transactions — fractional buys and redemptions, frequent rebalancing, small holders entering and exiting — that are economically impossible on a chain with high or volatile fees. Solana sustains on the order of 65,000 transactions per second, reaches finality in roughly 400 milliseconds, and settles transactions for approximately $0.00025 each. At that cost and latency, a holder redeeming a small fraction of a tokenized asset pays a negligible fraction of the trade in fees, and the market can support the transaction volume that continuous two-way liquidity implies. Every element of protocol state — supply, reserve, curve parameters, fees, and every historical transaction — is written on-chain and independently verifiable on Solana Explorer.

The engine exposes a small, deliberately constrained instruction set. Each client deployment is assembled from these primitives:

- create curve — establish a new bonding curve with its price function, base asset, and fee parameters fixed at creation.
- initialize bonding — set up the reserve and token accounts and bring the curve into a live, tradeable state.
- buy — deposit base asset, mint tokens against the curve, and advance the supply and reserve along the invariant.

- sell — burn tokens, return base asset from the reserve, and move the supply and reserve back down the same curve.
- update — adjust the permitted operational parameters of a live curve within the constraints fixed at creation.

Fees are handled entirely on-chain and symmetrically. Each curve carries a platform fee and a configurable issuer fee, both set at creation and thereafter visible on-chain to anyone. The same fee schedule applies to buys and to sells — there is no asymmetry that would penalize exit relative to entry. Crucially, fees are computed and distributed before the reserve state updates, so the invariant is always maintained on the net reserve: the amount that enters or leaves the reserve is the post-fee amount, and the curve is solved against that. This ordering is what allows fees to exist without ever making the reserve inconsistent with the supply. TokSol does not publish universal fee percentages, because there are none — the exact platform and issuer parameters are defined per engagement, calibrated to the asset and the client, and then frozen on-chain at creation.

6. The reserve as protocol property

The solvency theorem is only meaningful if the reserve cannot be removed by anyone other than a redeeming holder. This is enforced not by policy but by the absence of any instruction that would permit it. In a TokSol deployment there is no code path through which the base asset leaves the reserve except a holder calling sell and burning tokens against the curve. Any administrative capability to transfer or sweep reserves — the kind of function that appears in many token contracts and is the vector for the majority of exit-scam losses in the industry — is removed from the program before deployment. It is not disabled behind a flag; it is not present in the deployed bytecode.

The consequences are concrete. The token issuer cannot defraud buyers by draining the pool, because there is no instruction that lets the issuer touch the reserve; the issuer's only relationship to the reserve is that redemptions flow out of it to holders. The platform — TokSol — cannot defraud users, because the platform holds no privileged withdrawal path either. And a compromise of the web application or front end cannot reach the on-chain funds: a hijacked interface can mislead a user about what they are signing, but it cannot manufacture an on-chain instruction that does not exist, and the instruction to extract the reserve does not exist. The attack surface that dominates real-world crypto losses — privileged withdrawal, upgradeable drain functions, custodial compromise — is engineered out.

Because all of this is on-chain, solvency is auditable by anyone in real time. The reserve balance and the circulating supply are public; the price function and its parameters are public; the invariant relating them is public. Any observer can independently confirm that the reserve equals the integral of the price function over the current supply, at any block, without trusting the issuer or the platform. Solvency is not attested — it is checked.

7. Applying the engine to real-world assets

TokSol does not ship a single product. For each client we scope and deliver a bespoke platform, a bespoke protocol, and a custom bonding curve fitted to the specific asset. The engagement begins by studying the underlying: its valuation dynamics, the expected size and cadence of holder activity, the jurisdictions of the participants, and the legal form of ownership. From that study we calibrate the curve — the exponent and shape of the price function, the base asset, the fee structure, and the supply parameters — and we build the platform and protocol around it. The engine is proven; the calibration and the surrounding system are engineered per asset.

Representing an off-chain asset on-chain raises questions that the curve alone does not answer, and a serious deployment must address each of them explicitly:

- Legal wrapper and title — the legal vehicle that holds the underlying asset and the mechanism by which token holders have a defined, enforceable relationship to it.
- Custody and attestation of the underlying — who holds the physical or financial asset, and how its existence and condition are attested and evidenced to token holders.
- Valuation and oracle design — how the off-chain value of the asset is measured, how often, and how (if at all) that value is brought on-chain, with attention to manipulation resistance.
- Redemption and enforceability — what a token entitles the holder to claim against the underlying, and through what legal process that claim is enforced.
- Transfer restrictions — controls on who may hold or transfer tokens where a jurisdiction requires them, implemented at the protocol level when necessary.

We are deliberately honest about the boundary of what the protocol guarantees. The mathematical guarantees in this paper — permanent two-way liquidity and provable solvency — apply to the token and its on-chain reserve. They do not, and cannot, guarantee the enforceability, condition, or value of the off-chain asset. The curve ensures that a holder can always redeem their token for base asset at the curve price; it says nothing about whether the building still stands, whether the borrower repays, or whether a court in a given jurisdiction will honour the legal wrapper. Those risks live in the legal and operational design of each deployment, and they are the client's asset risk, not a property the engine can remove. Conflating the on-chain guarantee with a guarantee about the underlying would be dishonest, and we do not make that claim.

8. Risk considerations

A precise account of the guarantees requires an equally precise account of the risks. The following are material and are not eliminated by the protocol.

- Price risk — the curve price moves with supply, and the price at which a holder can redeem may be materially lower than the price at which they bought. The protocol guarantees the ability to transact, not the level of the price.
- Relative-liquidity risk — this is the distinction to internalize: liquidity is guaranteed, but the price at exit is not. A holder can always sell into the curve, but a large redemption moves down the curve and receives a lower average price than a small one. Guaranteed exit is not guaranteed exit value.
- Issuer and asset reputational risk — the value that the market ascribes to a token depends on confidence in the issuer and the underlying asset. Reputational damage, mismanagement of the underlying, or loss of confidence can depress demand and therefore the curve price, independent of any protocol behaviour.
- Smart-contract risk — the programs are software. While the instruction set is small and the reserve is protected by design, no non-trivial software is provably free of defects, and audits reduce but do not eliminate this risk.
- Custody risk — near-absent by design. Because no instruction can move the reserve except a redeeming holder, and no privileged withdrawal path exists, the custodial-compromise risk that dominates the sector is engineered close to zero for the on-chain reserve. This does not extend to custody of the off-chain underlying, which is a separate operational risk.

The honest summary is that TokSol removes a specific and historically devastating class of risk — the reserve cannot be stolen or drained through the protocol — while leaving price risk, asset risk, and residual software risk fully in place. Any participant should size their exposure with those residual risks clearly in view.

9. Regulatory considerations

This section is neutral and explicitly does not constitute legal advice. It describes considerations, not conclusions, and no party should act on it without independent counsel.

A token sold through a bonding curve without any issuer promise of profit is designed to sit outside the characterizations that typically capture securities. The classic tests turn substantially on an expectation of profit derived from the efforts of others; a curve that merely prices a token against supply, with no representation by the issuer about returns, is structured to avoid supplying that expectation. But structure is not dispositive. Classification depends on the jurisdiction and, heavily, on marketing framing — the same instrument can be positioned in a way that reintroduces the very expectation the mechanism was designed to avoid. How a token is described and sold can matter as much as how it is coded.

Real-world-asset-backed instruments raise a second layer. Depending on the nature of the underlying and the jurisdiction, the token may implicate securities law, commodity regulation,

collective-investment rules, or asset-specific regimes that apply regardless of the curve's design. A tokenized fund interest, a tokenized loan, and a tokenized commodity may each attract different and additional obligations. For these reasons, independent legal counsel — competent in each relevant jurisdiction — is required before any commercial operation. TokSol engineers the protocol; it does not opine on the legal classification of any client's instrument, and nothing in this paper should be read as such an opinion.

10. Conclusion

The tokenization of real-world assets has a well-documented failure mode: assets are issued into markets that never form, and holders are left with certificates they cannot sell. TokSol's answer is to make liquidity a property of the instrument rather than a hope about its market. A bidirectional cryptographic token prices both entry and exit against a base asset along a single continuous curve, so every holder can always transact against the protocol itself, with no counterparty to find and no external liquidity provider to depend on.

The reserve that funds redemptions is not a promise but a mathematical consequence — the integral of the price function over the circulating supply — and it is protected by the absence of any instruction that could remove it other than a redeeming holder. Solvency is therefore continuously auditable rather than merely asserted. Around this proven engine, TokSol builds a bespoke platform, protocol, and custom curve for each client asset, and is explicit that the on-chain guarantee covers the token and its reserve, not the enforceability of the underlying. The result is not a promise of return. It is a settlement mechanism for real-world assets in which the ability to exit is guaranteed by arithmetic — which, for assets that would otherwise die of illiquidity, is the property that matters most.

11. References

[01] Buterin, V. "On Path Independence" and related writing on constant-product and automated market makers, vitalik.eth.limo.

[02] Grith, M., Härdle, W. K., and Park, J. Shape-invariant modelling of pricing kernels and related nonparametric methods, Humboldt University / SFB 649 discussion papers.

[03] Zargham, M. and collaborators, BlockScience. Token bonding curve design and state-space / dynamical-systems treatment of bonding curves; cadCAD simulation lineage.

[04] Solana Foundation. Solana Documentation, docs.solana.com — architecture, throughput, finality, and fee model.

[05] Anchor. The Anchor Framework Documentation, anchor-lang.com — Solana program development in Rust.

[06] Boston Consulting Group. "Relevance of On-Chain Asset Tokenization in 'Crypto Winter'" — institutional projection of the tokenized real-world asset market.

[07] Standard Chartered. Research on the growth and adoption trajectory of real-world asset tokenization.

[08] McKinsey & Company. Analysis of tokenized financial assets and their projected adoption over the coming decade.