VOLTA: Trading Realized Volatility On-Chain

Abstract

VOLTA is a protocol for trading realized volatility as a native on-chain primitive. The system implements dual liquidity pools that automatically redistribute collateral based on calculated changes in exponentially weighted moving average (EWMA) volatility. In contrast to existing derivatives protocols that require users to express volatility views through options strategies, VOLTA allows direct exposure to volatility itself through fungible ERC-20 tokens.

The protocol introduces several novel mechanisms: self-contained synthetic leverage without liquidators, path-dependent token pricing based on volatility trajectories, and automatic pool rebalancing triggered by high-frequency oracle updates. These features are implemented as a single set of non-upgradeable contracts deployed on HyperLiquid.

1 Introduction

Volatility derivatives allow market participants to take positions on the magnitude of price movements rather than price direction. In traditional finance, these instruments include variance swaps, VIX futures, and volatility indices that collectively represent hundreds of billions in notional value.

Current decentralized finance protocols provide limited volatility exposure. Options protocols require users to construct complex strategies (straddles, strangles) across multiple strike prices and expiration dates to approximate volatility positions. These constructions suffer from time decay, require active management, and provide imperfect volatility exposure due to their dependency on directional price movements.

VOLTA implements a mechanism for direct volatility exposure through dual liquidity pools and deterministic rebalancing. The protocol calculates realized volatility using exponentially weighted moving averages and automatically redistributes collateral between opposing pools based on volatility changes. This design enables users to take long or short positions on volatility through simple token minting and redemption.

2 Dual Pool Architecture

The core mechanism of VOLTA consists of two USDC liquidity pools representing opposing volatility positions:

VOL(+) Pool: Benefits when realized volatility increases.
VOL(-) Pool: Benefits when realized volatility decreases.

Each pool issues fungible ERC-20 tokens (VOL+ and VOL-) that represent proportional claims on pool collateral. Token redemption value is calculated as:

$$\text{TokenPrice} = \frac{\text{TotalPoolTokenCollateral}}{\text{TotalPoolTokenSupply}} \tag{1}$$

Users mint tokens by depositing USDC collateral into the corresponding pool. The minting ratio is determined by the current collateral-to-supply ratio at the time of deposit.

2.1 Volatility-Based Rebalancing

On each oracle update, the protocol computes a transfer amount proportional to the volatility delta, scaled by the pool that pays out. To ensure that the transfer ratio remains bounded between 0 and 1, this raw signal is passed through the Gaussian error function:

$$\Delta C = \text{erf}\Big(k \cdot \Delta rVol\Big) \cdot \text{Pool}_{\text{from}}, \tag{2}$$

$$\text{Pool}_{\text{from}} = \begin{cases} \text{Pool}_{-}, & \Delta rVol > 0 \quad \text{(volatility rises; payout from VOL}- \text{ to VOL}+\text{)}, \\ \text{Pool}_{+}, & \Delta rVol < 0 \quad \text{(volatility falls; payout from VOL}+ \text{ to VOL}-\text{)}, \\ 0, & \Delta rVol = 0\ . \end{cases} \tag{3}$$

Here:

- $\Delta rVol$ is the realized volatility change since the last update,
- k is a fixed coefficient controlling sensitivity,
- $\text{erf}(\cdot)$ ensures that the effective multiplier is always in [0,1).

This formulation guarantees that transfers are capped by available collateral in the paying pool while avoiding pathological jumps for large volatility moves. It also preserves asymmetric return profiles when pools are imbalanced.

3 Oracle Integration and Volatility Calculation

3.1 HyperLiquid Oracle System

Volta leverages HyperLiquid's native price oracle, which aggregates spot prices from major centralized exchanges and updates every three seconds.

The protocol queries the oracle at the beginning of each user interaction and recalculates volatility before executing any state changes. This ensures that all operations reflect the most current market conditions.

3.2 EWMA Volatility Calculation

Realized variance is updated continuously using a smoothing factor that depends on the elapsed time between oracle updates. Specifically,

$$\text{var}_t = \text{var}_{t-1} \cdot e^{\alpha\,\Delta t} + \Big(1 - e^{\alpha\,\Delta t}\Big) \cdot \left(\ln \frac{P_t}{P_{t-1}}\right)^2, \tag{4}$$

where

- P_t is the asset price at the current oracle update,
- Δt is the elapsed time since the previous update,
- $\alpha < 0$ is the continuous-time smoothing rate,
- var_t is the exponentially weighted variance at time t.

Realized volatility is then defined as

$$rVol_t = \sqrt{\text{var}_t}. \tag{5}$$

Note. Because $\alpha < 0$, the factor $e^{\alpha \Delta t}$ always lies in $(0, 1)$ and acts as a *decay weight* on the previous variance. This convention eliminates the need to write an explicit negative in the exponent while ensuring exponential smoothing behaves correctly. In the special case where Δt is constant, this formulation reduces to the familiar discrete-time EWMA with a fixed decay parameter $\lambda \in (0, 1)$.

Recent price movements receive higher weighting than historical data, providing responsive volatility measurement that adapts to changing market regimes.

4 Anti-Front-Running Design

Every contract function begins with an oracle query and a volatility recalculation, so user interactions are always priced against data that is only one oracle tick old.

The oracle updates on a fixed three-second interval, and when combined with the mandatory transaction fee on each interaction, this creates strong economic disincentives for front-running. An attempted attack—such as buying immediately before a pool payout is applied—will yield less profit than the fee required to execute the trade, resulting in negative expected value for would-be front-runners.

5 Synthetic Leverage Mechanism

5.1 Leveraged Token Issuance

Users can mint leveraged positions by depositing collateral and selecting a leverage factor. The number of tokens minted is given by

$$\text{Tokens}_{\text{minted}} = \frac{\text{Collateral}_{\text{deposit}} \cdot L}{P}, \tag{6}$$

where

- $\text{Collateral}_{\text{deposit}}$ is the amount of stablecoins deposited,
- L is the chosen leverage factor (e.g. 2x, 3x),
- P is the current token price determined by the protocol's settlement function.

Any tokens beyond the 1x equivalent are escrowed to back the leveraged exposure. The protocol monitors the escrow continuously and burns the excess tokens if the collateral buffer is exhausted.

5.2 Automatic Position Management

Leveraged positions are monitored continuously. Whenever a user opens or updates a position, the corresponding liquidation price is precomputed based on the collateral deposited, the leverage factor, and the escrowed tokens. If the market reaches this liquidation price, the contract executes liquidation automatically:

- Burns escrowed tokens
- Seizes remaining collateral
- Closes the position

This eliminates the need for liquidators, auction mechanisms, or manual intervention. The maximum loss is predetermined and bounded by the initial collateral deposit.

Leveraged positions on VOLTA are closed automatically when the collateral buffer breaches the protocol threshold (see Sections 4.2–4.3). Checking every position against an exact trigger on each oracle update is costly. Instead, VOLTA discretizes liquidation thresholds into *buckets* and evaluates only the buckets that are crossed when the oracle updates. This keeps on-chain work predictable while preserving user fairness.

5.3 Liquidation Conditions

A leveraged position becomes subject to automatic liquidation when its collateral buffer is fully depleted relative to the escrowed exposure. Formally,

$$\text{Collateral}_{\text{buffer}} \;\leq\; 0, \tag{7}$$

at which point the protocol burns the escrowed tokens and seizes the posted collateral.

Note. Unlike traditional margin systems, there is no minimum collateralization ratio or liquidation threshold parameter. Positions either remain open while their buffer is positive, or are closed immediately once the buffer is exhausted. This design eliminates cascading liquidations and simplifies the closure logic.

5.4 Bucket Definition

Let x denote the liquidation trigger (e.g., a price or health ratio) expressed in the contract's fixed-point units. We define a bucket function

$$B_k(x) \;=\; \lceil x \text{ rounded to its first } k \text{ significant binary digits} \rceil$$

that applies a ceiling to x, zeroing the remaining bits upward. All positions whose trigger maps to the same $B_k(x)$ are indexed into that bucket.

Intuition. The bucket width scales with the magnitude of x, yielding approximately constant *percentage* precision across the operating range while remaining cheap to compute via bitwise operations. By applying a ceiling rather than a symmetric rounding rule, the computed liquidation price is always slightly higher than the true threshold by at most one bucket's width. This ensures the protocol never liquidates too late and avoids losing collateral to dust. Users may therefore be liquidated marginally early, but never late.

Rounding Mode. VOLTA adopts a *ceiling function* rather than round-to-nearest. This guarantees $B_k(x) \geq x$, ensuring that liquidation always occurs at or slightly above the true liquidation price.

Intuition. The bucket width scales with the magnitude of x, yielding approximately constant *percentage* precision across the operating range while remaining cheap to compute via bit operations. On each oracle tick the contracts (i) compute the current trigger, (ii) determine which buckets were crossed, and (iii) process only those buckets. Oracle ticks precede any state changes.

5.5 Dust and Accounting

Rounding a trigger to $\mathsf{B}_k(x)$ creates a small delta (*dust*) between the bucket boundary and the exact trigger. VOLTA tracks dust explicitly and refunds the dust ont he next interaction.

6 Path Dependency and Pool Dynamics

6.1 Token Price Behavior

VOL+ and VOL- token prices exhibit path-dependent behavior driven by three variables:

- Pool collateral balance,
- Outstanding token supply,
- Realized volatility trajectory.

Two periods with identical starting and ending volatility levels can still produce different token returns because of how collateral flows interact with the current token supply. This path dependency emerges naturally from the continuous rebalancing mechanism and results in non-linear return profiles for both VOL+ and VOL.

6.2 Pool Imbalance Effects

When pools become imbalanced in size, volatility changes have disproportionate effects on pool gains. For example, if the VOL+ pool contains significantly less collateral than the VOL- pool, a given increase in volatility will result in larger percentage gains for VOL+ token holders. This asymmetry is governed by the transfer function in Section 2.1, which uses the minimum pool balance as a scaling factor.

7 Fee Structure

The protocol implements a fee structure designed to generate revenue while discouraging exploitative behavior.

The absence of redemption and liquidation fees eliminates the need to adjust liquidation thresholds to account for exit costs, which would otherwise effectively increase leverage ratios.

Fee Type	Rate	Application
Minting Fee	30 bps	Applied to USDC deposits
Redemption Fee	0 bps	No fee on token redemptions
Liquidation Fee	0 bps	No fee on automatic liquidations
Leverage Funding Fee	[TBD]	Applied to leveraged positions
Adjustment Fee	[TBD]	Applied during pool rebalancing

Table 1: Protocol Fee Structure

8 Risk Considerations

8.1 Oracle Risk

The protocol's reliance on HyperLiquid's oracle creates dependency on the underlying price feed accuracy and availability. Oracle manipulation or failure could affect volatility calculations and pool rebalancing.

8.2 Pool Depletion Risk

The use of the error function when calculating adjustment amounts prevents either pool from being drained entirely.

Disclaimer

This paper is for informational purposes only and does not constitute investment advice. The protocol involves significant risks including total loss of deposited capital. Users should carefully consider their risk tolerance and conduct their own research before interacting with the protocol.