Design of the Gyroscope Consolidated Price Feed and Circuit Breaker System

Ariah Klages-Mundt*

September 2022

1 Introduction

While DeFi protocols exist as self-executing financial contracts on-chain, their execution often relies on off-chain information, such as prices of reference assets. Oracles are data feeds that provide *estimates* of this information in an on-chain format. Since the processes that create this information do not live on-chain, the correctness of an oracle's estimate cannot in principle be fully verified on-chain. Herein lies the blockchain oracle problem: reliance on oracles can re-introduce trust assumptions and counterparty risks into protocols that otherwise aim to remove these risks.

Prior oracle systems generally follow one of the following models, which each present significant challenges.

Centralized oracles require trust in a central data provider. Protocols that use centralized oracles bear counterparty risk from the provider. The provider's data may be incidentally wrong, the provider may go offline, or the provider may behave dishonestly should the reward from supplying manipulated data be more profitable than providing true data. Using a centralized oracle has the security model of involving a trusted third party.

Medianizing of several centralized nodes. These aggregate data from a number of oracle nodes in various ways, a threshold of which must then be incorrect in order for the medianizer to report incorrect data. Note that we can't verify that the medianizer's output is correct, it can only be verified that it is the desired median computation. As a security model, using a medianizer has the security model of involving a trusted multisig. In variations, nodes can be slashed for providing data different from the median. This faces similar incentive issues as Schelling point games discussed below in betting market designs.

DEX TWAPs. Prices that occur on decentralized exchanges (DEXs), such as Automated Market Makers (AMMs), can be used as price oracles in some situations. Making DEX

*Researcher at Superluminal Labs, a software development company working in the Gyroscope ecosystem. In a separate capacity to this work, the author is a PhD student at Cornell University.

price measures resilient is challenging: one has to account for limited liquidity and time average over extended time periods to make manipulation sufficiently costly. DEX time-weighted average prices (TWAPs) can be exploited when the manipulation costs are not high enough, as quantified in [1] and as observed in [2]. However, since DEX trades occur on-chain, these prices can be observed directly on-chain and the costs to manipulate them are also quantifiable on-chain. They have two further disadvantages: TWAPs are inherently *slow* to react to market changes as they involve observing over a trailing period of time, and DEXs can only represent prices in terms of on-chain assets. On the second point, the prices that are usually of interest are in terms of national currencies like USD, which can't be represented in this manner. Prices in terms of other stablecoins (like USDC) can be used as a proxy. However, this setup faces the same oracle problems: the protocol then relies on that stablecoin (and often a centralized issuer), which may be manipulated or fail, for the integrity of the data feed. It moves the problem as opposed to solves it.

Betting markets. Other decentralized approaches exist that rely on Schelling point games, in which agents vote on the data feed and are incentivized by slashing if their vote deviates from the consensus. Since the consensus is not objectively verifiable for correctness and the incentives are related to it, these are problematic and can be manipulable through game theoretic attacks. Essentially, an agent's incentives can be skewed to vote based on what they believe the output of the consensus will be as opposed to voting based on what they know the truth to be (this is a form of beauty pageant problem in economics).

1.1 Our approach

We take a different approach to designing an oracle system. While correctness of off-chain prices cannot be verified on-chain in principle, the *likelihood* of correctness given other observable variables can be assessed on-chain.

We present a new approach, which we call the *Consolidated Price Feed* (CPF), for hardening blockchain price feeds by consolidating information from various on- and off-chain sources and validating the likelihood of information integrity on-chain. The CPF optimistically achieves the speed of Chainlink, which is an oracle that goes through a medianizing process off-chain, but with fallback guarantees if Chainlink prices are wrong for whatever reason. The CPF minimizes reliance on trusted price feeds by being able to evaluate data integrity on-chain and automatically detect likely pricing failures and provides guardrails against known and unknown oracle risks.

The CPF system was designed with several design aims:

1. Typically, the system achieves the speed of a centralized price feed [Aim 1]
2. The system is live (i.e., functional for querying prices) in most settings but errs on the side of safety when data integrity is less certain [Aim 2]
3. An agent who tries to attack the system by manipulating oracle prices faces high costs, in a way that is quantifiable on-chain, and uncertain effectiveness [Aim 3]
4. The system is able to detect when provided prices may be unintentionally stale [Aim 4]

5. An agent who tries to cause the system to not be live for an extended time period (e.g., Denial of Service) faces a high cost that is quantifiable on-chain [Aim 5]
6. The system is able to price tokenized LP strategies on underlying tokens (in particular, LP shares in AMMs) while achieving the same robustness properties [Aim 6]

We achieve these aims through three pieces of infrastructure:

- A *consolidation mechanism* that determines a likelihood of a set of price estimates being correct by triangulating with other sources of information on-chain.
- A sequence of *circuit breakers* that pause the system when state or changes to state trigger large uncertainty about oracle or wider system security.
- A library of Liquidity Provider (LP) share pricing formulations that are resistant to manipulation and market changes.

Note that this oracle design was developed for Gyroscope's use case in the Dynamic Stability Mechanism (DSM), which facilitates the minting and redemption capability of Gyroscope stablecoins. Other types of applications may have different oracle design aims, in particular around liveness vs security prioritization. While pausing the DSM for short periods of time would have low impact on the system overall, other systems may need to perform fast actions even if oracle-provided prices have less certainty. In such cases, the CPF could be adapted to simply raise red flags, in which case protocols could be encoded to respond more conservatively given less certain oracle information. Or alternatively, the CPF architecture could be adapted to estimate prices as accurately as possible (e.g., through filtering) as opposed to pausing when price information is inconsistent.

2 Data Sources and Security Risks

Let S be a set of on-chain assets referenced by a DeFi protocol. Consider that ETH is an asset in S and that it is has high market liquidity among all assets in S.[1] Denote the true market prices as $p_{i/j}$ for the price of asset $i \in S$ in terms of asset $j \neq i \in S \cup \{\text{USD}\}$.

The CPF uses data from the following sources, which provide either estimates of true prices or other information that can indicate if price estimates are wrong.

Chainlink. For $i \in S$, denote the Chainlink-provided prices as $\hat{p}_{i/\text{USD}}$. The security model of Chainlink amounts to a large but trusted multisig and a trusted relay. Many Chainlink nodes report prices, which go through an off-chain medianizer and then relayed on-chain [6]. Chainlink has a rigorous off-chain infrastructure for aggregating price data and the speedy relay of prices on-chain. There is no current functionality to *prove* that the data relayed on-chain was computed as the correct median of prices reported from nodes, which is why the relay is trusted. Although there is a reputation system for nodes with a proposed incentive system involving slashing [3], the system still effectively relies on a trusted multisig to enforce the alignment between consensus price and true price.

[1] Note that if this is not true, another asset can be substituted for ETH.

AMM TWAPs. For $i \neq j \in S$, denote the TWAP price as $\bar{p}_{i/j}$. In reality, there are many TWAP prices possible both in terms of duration of the TWAP and market (e.g., Uniswap, Balancer, Curve). For simplicity, we take a single TWAP for each relevant pair as defined and later discuss how the definition can be calibrated. These TWAPs do not require trust, as they exist on-chain and do not require a relay. However, since they are manipulable and estimate off-chain prices slowly, their use in any application should require caution and study of these effects. Recall, further, that TWAPs do not include USD prices without relying on a trusted anchor for that price (e.g., relying on Circle to maintain USDC at \$1).

Exchange-signed prices. Consider that off-chain exchanges numbered $0, \ldots, n$ provide signed prices and denote the ETH/USD signed price for exchange k as $\tilde{p}^k_{\text{ETH/USD}}$. Since the signatures for these prices can be verified on-chain, the relay for these prices does not need to be trusted, though the data provider (i.e., the exchange) is a trusted counterparty. For example, Coinbase and OKX provide signed prices [4, 7].

Observing changes to protocol state. Let $f^{\text{IN}}_{x,t}$ be the flow of assets $x \in S$ occurring at transaction sequence t (e.g., the order of transactions on the blockchain) moving into the protocol. And similarly let $f^{\text{OUT}}_{x,t}$ the flow moving out of the protocol. Let $b(t)$ give the block number that contains transaction sequence t. For $x \in S$, $k \in \{\text{IN}, \text{OUT}\}$, and $T \geq 0$ a position in the transaction sequence, define the block-discounted exponential moving sum of a flow as

$$F^k_{x,T} = \sum_{t \leq T} \delta^{b(T)-b(t)} f^k_{x,t},$$

where $0 \leq \delta \leq 1$ is the memory parameter of the moving sum. For a large memory parameter, the exponential moving sum retains more information about flows further into the past. For a small memory parameter, it only retains information about recent flows. This information about how agents interact with the protocol is completely observable on-chain with no trust assumptions. Unusual behavior can be used to flag suspicious settings when oracle prices may be wrong.

This measure is implemented efficiently considering that we do not need to store the past flow amounts. It is only required to store the last computed value and the block in which it was last updated. Suppose we are at T in the transaction sequence and the previous transaction with the protocol took place at point $T' < T$ in the transaction sequence. Then the value of $F^k_{x,T}$ can be computed as

$$F^k_{x,T} = \delta^{b(T)-b(T')} F^k_{x,T'} + f^k_{x,T}$$

Note that the formula is correct whether T' and T occur in the same block or not (i.e., whether $b(T') = b(T)$ or $b(T') < b(T)$).

The CPF is a general framework and can incorporate other alternative data in the future to further strengthen oracle guarantees. For example, future data sources might include the following:

- Succinct proofs that Chainlink relayed prices are medianized off-chain from prices signed by Chainlink nodes (should Chainlink add this functionality).

- Data on other on-chain economic systems in which agents' decisions incorporate information about the prices of assets. For example, on-chain lending and derivative markets, PoS staking mechanisms, and gas fee levels encode some information about the ETH/USD price since agents incorporate this price into their actions. As explored in [8], methods can be used to extract signal on the ETH/USD price from the wider noise in ways that can help to improve oracle security.

3 Consolidation Mechanism

The CPF consolidation mechanism cross-references different types of price information from different sources to determine a likelihood that a set of prices is correct. It takes as input the prices $\hat{p}_{i/\text{USD}}$, $\bar{p}_{i/j}$, and $\tilde{p}^k_{\text{ETH/USD}}$ for $i \neq j \in S$ and flags whether the set of prices $\{\hat{p}_{i/\text{USD}} | i \in S\}$ is likely to be correct or not given the consistency of the information available. The mechanism can be calibrated to have high manipulation costs and high Denial of Service (DoS) costs.

The consolidation mechanism draws inspiration from the design of the Compound Open Price Feed [5], which incorporates a `UniswapAnchoredView`, while adding a new structure to consolidating information. In the Open Price Feed, the USD price of an asset is reported by an external oracle and, as a sanity check, is not allowed to deviate too far from the USDC price of that asset reported by a TWAP on Uniswap. There are a few cases in which this structure for cross-referencing information can fall short:

- Most importantly, it relies on the assumption that 1 USDC = 1 USD. In general, this may not be true, and a robust mechanism would need to correct for deviations.
- When a protocol wants to price a set of assets at once (e.g., the set S), information gained about one asset can be used to improve the level of information about different but connected assets. As a result, it can be more optimal (e.g., in terms of likelihood ascertained per gas expended) to design a mechanism that jointly consolidates pricing information on the assets in S as opposed to one that consolidates information on individual assets at a time.
- When the oracle reported USD price is detected as inconsistent with the Uniswap USDC price (i.e., outside of the range of allowed deviation), the Open Price Feed design reports the nearest price that is within the range of allowed deviation as opposed to waiting some time until prices can be better verified. Effectively, this is a design choice to favor oracle liveness at the expense of security, which may not be the right choice for all applications.

The CPF consolidation mechanism is designed to address these points to provide a more robust way to consolidate different types of pricing information. It combines two types of checks that cross-reference asset prices:

- *Relative price checks* compare information about the price of one asset in S relative to another asset in S. Observing DEX markets is well-suited for this as these pairs can trade on DEXs.

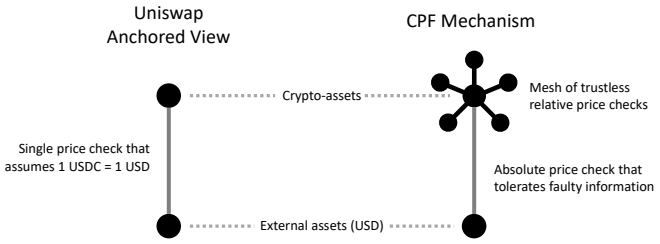

Figure 1: High level consolidation structure of CPF vs Uniswap Anchored View.

- *Absolute price checks* compare information about the price of an asset in S relative to external assets (e.g., USD). DEXs alone are not well-suited for this as these pairs cannot trade on DEXs.

The joint structure of these checks forms a web among DEX asset pairs in S, grounded by cross-referenced information about USD prices and can be calibrated in optimal ways. The mechanism that we implement is relatively simple, involving threshold checks, while the structure captures the nuances of different price information.[2]

The CPF consolidation mechanism is compared to the Uniswap Anchored View mechanism in Figure 1 in terms of high level structure of price checks.

3.1 Relative price checks

A relative price check between two assets $i, j \neq i \in S$ takes as input $\hat{p}_{i/\text{USD}}, \hat{p}_{j/\text{USD}}$ and $\bar{p}_{i/j}$. It cross-references the information based on the identity (in terms of true prices)

$$p_{i,j} = \frac{p_{i/\text{USD}}}{p_{j/\text{USD}}}.$$

Given $\varepsilon_{i,j} > 0$, a relative price check returns true if

$$\frac{\left|\bar{p}_{i/j} - \frac{\hat{p}_{i/\text{USD}}}{\hat{p}_{j/\text{USD}}}\right|}{\bar{p}_{i/j}} \leq \varepsilon_{i,j}$$

and returns false otherwise. That is, the relative price check checks whether $\bar{p}_{i/j}$ is sufficiently close to the comparable price inferred from $\hat{p}_{i/\text{USD}}, \hat{p}_{j/\text{USD}}$.

For example, consider that we have the two assets ETH and WBTC. Chainlink gives us price estimates for ETH/USD and WBTC/USD, from which we compute the corresponding price estimate of ETH/WBTC using the identity. The relative price check then compares the ETH/WBTC price implied by the Chainlink estimates to the ETH/WBTC price estimated from an AMM TWAP.

Considering that there are n assets to price in S, there are (n choose 2) possible pairs to choose for relative price checks. Choosing more pairs adds to overall manipulation costs because more pools need to be manipulated in an attack. The manipulation costs scale by

[2]Other more advanced checks are possible, though more involved to implement on-chain.

the number and liquidity of DEX pools referenced. There is a trade-off, however, as choosing more pairs also increases the gas expended in computing the oracle and can also reduce the cost of a DoS attack. To perform a DoS attack, an attacker needs to cause one (or in a more general implementation, some threshold number) of relative price checks to fail, which faces the tangible cost of manipulating the pool(s) with the least liquidity relative to the threshold ε. Adding pools that have lower liquidity over a broad pricing range thus reduces the cost of a DoS attack.

The costs to manipulate an AMM pool can be estimated using methods like in [1]. The manipulation costs are related to the size of assets that need to be committed to push the pool to an unbalanced state (e.g., creating an arbitrage opportunity) and the probability that this pool is not rebalanced within the same block and subsequent blocks. Since these manipulations cannot be achieved with flash loans, the manipulation cost is related to the risk that an attacker takes on in committing assets to the strategy considering that the manipulation may not be successful.

TWAPs may be problematic in pools with highly concentrated liquidity since the spot price at the beginning of each block (which goes into the TWAP calculation) only has high manipulation costs up to the thresholds of high liquidity in a pool. Beyond the region of high liquidity, the pool spot price can be manipulated almost arbitrarily high with little cost. For this reason, standard TWAPs are only robust when the pool has high liquidity across a wide price range.[3]

Given information on the liquidity of pools, marginal costs of on-chain computation, and bounds on the acceptably high costs of manipulating the oracle feed and performing a DoS attack on the mechanism, the structure of the optimal 'mesh' of relative price checks (i.e., which asset pairs to choose) can in principle be solved as an optimization problem. In practice, a 'hub and spoke' mesh, in which pairs are chosen with common central assets like ETH, are likely good heuristics.

When well-calibrated, the mesh of relative price checks validate the consistency of relative prices coming from an external feed like Chainlink, but it does not necessarily validate *absolute* prices with respect to external assets (e.g., USD prices). For example, externally reported USD prices could pass relative price checks if all absolute prices were scaled by 2, as this scaling cancels out in the relative price measures. The mechanism needs the next type of absolute price checks to do this. Given the mesh of relative price checks, however, it is enough to do absolute price checks on a single asset.

3.2 Absolute price check

In the absolute price check, the price of an asset $i \in S$ in terms of an external asset (e.g., USD) is checked for consistency with several independent data sources. This effectively 'grounds' the mesh of relative price checks. Note that while relative price checks can be done without third party trust assumptions, as they only use on-chain DEX information /footnotealthough relative price checks *do* require proper understanding of market manipulation costs, the absolute price check does involve information that is not natively on-chain. Here the CPF mechanism must adopt a trust minimization approach, where this trust minimization can

[3]TWAPs could be designed to be more resilient to this issue if the estimates contributing to the TWAP were capped within the range of major liquidity provision, but this is not done in standard TWAPs today.

be demonstrated on-chain. The goal is to make the absolute price check as resilient to data corruption or manipulation as possible.

Suppose that the absolute price check is performed on ETH/USD. A different asset could be selected without changing the structure, but ETH will typically make sense because it is likely the most liquid asset over the widest price range and so will provide the strongest guarantees in the price check. The absolute price check takes as input $\hat{p}_{\text{ETH/USD}}, \tilde{p}^k_{\text{ETH/USD}}, \bar{p}_{\text{ETH}/b}$ for $k \in E$ and $b \in B$ where E is a set of external data providers (e.g., exchanges who provide signed prices) and B is a curated set of stablecoins. The set of stablecoins B needs to be curated to focus on stablecoins that are expected to keep peg (e.g., custodial ones) and have sufficient liquidity in DEXs.

Define m as follows:

$$m := \begin{cases} \texttt{minimum}\left\{\bar{p}_{\text{ETH}/b} | b \in E\right\} & \text{if } |E| \leq 2 \\ \texttt{secondMin}\left\{\bar{p}_{\text{ETH}/b} | b \in E\right\} & \text{if } |E| > 2 \end{cases}$$

That is, m is either the minimum or second minimum (i.e., second smallest element) DEX TWAP price for ETH-stablecoin pairs from the curated set B.

Define $\tilde{p}$ as follows:

$$\texttt{median}\left(\left\{\{\tilde{p}^k_{\text{ETH/USD}} | k \in E\} \cup \{m\}\right\}\right)$$

That is, $\tilde{p}$ is the median of the price information from sources E and the filtered information m.

Given $\varepsilon_{\text{ETH,USD}} > 0$, the absolute price check returns true if

$$|\hat{p}_{\text{ETH/USD}} - \tilde{p}| \leq \varepsilon_{\text{ETH,USD}}$$

and false otherwise.

The calculation is structured this way so that, in addition to independent external data feeds for ETH/USD (coming from E), the check can be strengthened by incorporating on-chain estimates of the ETH/USD price through m. The most straightforward on-chain estimate is to interpret the USDC price as 1 USD, which is equivalent to interpreting the issuer Circle indirectly as the oracle), and do similarly for other stablecoins. The issue with this interpretation is that different centralized stablecoins face similar risk factors (e.g., censorship and regulatory) and so are not really independent measures, and decentralized stablecoins are either expected to deviate from peg or simply maintain peg by pegging to a centralized stablecoin (e.g., Dai through its Peg Stability Module). The structure needs to account for this, which is why m uses a different medianizing method and why the set of stablecoins B needs to be curated (and possibly changed by governance over time).

The alternative medianizing method for m uses the second minimum because the stablecoins in B (e.g., custodial stablecoins) are possibly worth less than 1 USD but unlikely to be worth more than 1 USD. If B is large, it is likely that some remain pegged. The second minimum allows this information to be isolated in a way that is resilient to many stablecoins depegging below peg and one stablecoin trading above peg. This information in m is then incorporated into the set of external price feeds for ETH/USD, from which a median is taken.

Supposing $|E| = 2$, the median is resilient to all stablecoins in B depegging or Chainlink nodes colluding together and with another external oracle in E.

In the future, further on-chain information can be incorporated into the median to increase resilience further, e.g., measures signals on the ETH/USD price extracted from the behavior of agents on-chain economic systems, as explored in [8].

3.3 Implementation

Table 1 lists the most important functions that implement the above calculations. They are all defined in the file `CheckedPriceOracle.sol`.

Function	Purpose
`ensureRelativePriceConsistency`	Executes a single relative price check.
`batchRelativePriceCheck`	Executes the mesh of relative price checks.
`getRobustWETHPrice`	Executes the absolute price check.
`getPricesUSD`	Executes the consolidation mechanism including price checks.

Table 1: Most important functions for consolidation mechanism

Table 2 lists the parameters that need to be calibrated in the implementation. They are defined in the file `CheckedPriceOracle.sol`. Additionally, the current `relativeOracle` implementation is `UniswapV3TwapOracle.sol`; the TWAP pools used in the mechanism need to be registered there.

Parameter	Purpose
`usdOracle`	Primary oracle (e.g., calling Chainlink).
`trustedSignerPriceOracles`	Register of price oracles in E.
`assetsForRelativePriceCheck`	Quote assets used for relative price checks.
`quoteAssetsForPriceLevelTWAPS`	Register of stablecoins in B.
`relativeMaxEpsilon`	Deviation threshold for relative price checks.
`MAX_ABSOLUTE_WETH_DEVIATION`	Deviation threshold for absolute price check.

Table 2: Parameters for consolidation mechanism

Note that in the current implementation, the deviation thresholds for relative price checks is set as a common value for all relative price checks.

The following are important points in calibrating the consolidation mechanism:

- DEX pools for TWAPs need to have reasonable liquidity across a wide price range.
- Thresholds ε for price checks and time periods for TWAPs should balance requirements for liveness and safety based on the expected level of deviation of TWAPs.
- The set of external price feeds E should contain price feeds that are relatively independent from each other and, when they represent exchange-signed prices, the exchanges should have reasonable market depth.

- The set of stablecoins B should be curated to focus on stablecoins that are expected to keep peg (e.g., custodial ones) and have sufficient liquidity in DEXs. They should be chosen considering risk factor commonalities between the stablecoins in the set.

4 Circuit Breakers

The CPF also has a series of circuit breakers that are designed to protect the application protocol in case faulty oracle information makes it through the consolidation mechanism as well as providing protection against more general risks, such as mitigating the effects of smart contract bugs.

4.1 Flash crash circuit breaker

If oracle prices as reported by Chainlink change by more than a threshold in a given amount of time, then the flash crash circuit breaker initiates safety mode in the protocol. In particular, let $p_{x,t}$ be the Chainlink-reported price of asset x at time t, and let u_x and $\theta_{x,\text{flash}}$ be the corresponding time interval and threshold for asset x. Then the circuit breaker is triggered if

$$\frac{\left|p_{x,t} - p_{x,\lfloor t-u \rfloor}\right|}{p_{x,t}} \geq \theta_{x,\text{flash}},$$

where the floor function is redefined over the index set of Chainlink price report times as opposed to integers.

The flash crash circuit breaker is implemented in `CrashProtectedChainlinkPriceOracle.sol` with parameters `minDiffTime` and `maxDeviation` registered for each asset.

4.2 Excessive flow rate circuit breaker

The next circuit breakers initiate a *safety mode*, which is a pause of protocol operations over a set time period. For Gyroscope, the initial application is the Dynamic Stability Mechanism (DSM), which is an automated market maker that defines minting and redemption of stablecoins against reserve assets. Safety mode is defined in an application-specific way, i.e., pausing relevant features of a protocol and perhaps not all features.

The excessive flow rate circuit breaker is designed as a last resort to protect against oracle exploits as well as smart contract bugs and unknown exploits. In such events, this circuit breaker stops the bleeding automatically. It operates by measuring ultra-short term flows within the protocol and triggering safety mode if they would exceed thresholds. In principle, this circuit breaker can protect against an attacker who can arbitrarily manipulate oracle prices at a very short time scale. It prevents an atomic exploit from draining the protocol assets more than the threshold amount, and also severely limits the speed with which non-atomic exploits can be performed, which usually increases their cost and gives other mechanisms time to respond. Define the flow rate threshold for asset x as $\theta_{x,\text{flow}}$ and define a threshold buffer $0 \leq \varepsilon_{\text{flow}} \leq 1$. Informally, new flows will be disallowed if recent flows exceed $\theta_{x,\text{flow}}$ and safety mode will be triggered if recent flows exceed $\varepsilon_{\text{flow}}\theta_{x,\text{flow}}$.

Recall the measure of flows $F^k_{x,T}$ defined in 2 as a block-discounted exponential moving sum with memory parameter δ. Let δ be chosen so that it is sufficiently close to zero so that

the measure represents ultra-short term flows (this is a calibration decision). Now suppose we are at place T in the sequence of transactions and that the current transaction is of type $k \in \{\text{OUT}, \text{IN}\}$ in the protocol (i.e., it would initiate a flow $f^k_{x,T}$). Next calculate what $F^k_{x,T}$ would be if the flow was allowed. The excessive flow rate circuit breaker works in the following way. If

$$F^k_{x,T} > \theta_{x,\text{flow}}$$

then the flow is not allowed. Alternatively, if

$$\varepsilon_{\text{flow}}\theta_{x,\text{flow}} \leq F^k_{x,T} < \theta_{x,\text{flow}}$$

then the flow is allowed but safety mode is initiated for a time period on subsequent transactions with the protocol. If neither is triggered, the flow is allowed as usual.

For the Gyroscope DSM, the flows $f^k_{x,t}$ to focus on are the amounts of Gyroscope stablecoins minted (for $k = \text{IN}$) or redeemed (for $k = \text{OUT}$) in the transaction at place t. In other protocols, different definitions of flow may be better, but otherwise the same mechanism can be used.

DoS attack. In calibrating the circuit breaker, there is a DoS attack to consider. An attacker can max out the allowed short term flows on an asset iteratively over time by cycling assets through mint and redeem operations against that asset. This effectively causes the DSM to be shut down for the given asset market while the attack continues. To do so on all DSM markets, the attacker would need to trigger the max flows on all assets at the same time. Presumably the attacker would be motivated by an outside profit opportunity in disrupting the system. This DoS attack comes with sizable manipulation costs as the attacker would have to pay the spread between minting and redeeming in the DSM for all flows it puts through the system. Notably, the fees paid would go to bolster the protocol's asset reserves, which may be at least partially counterproductive to the aim of the attack.

The manipulation costs are related to system parameters, which can be tuned to make the DoS attack acceptably costly. For example, consider that the DSM spread between minting and redeeming for a given reserve asset x is 0.1%, which might make sense for instance if $x = \text{USDC}$. Let the threshold $\varepsilon_{\text{flow}}\theta_{x,\text{flow}} = 10\text{m}$. Suppose δ is chosen so that the exponential moving sum effectively decays over 12 blocks (2.4 minutes) and that the automatic safety mode is similarly for 12 blocks. Lastly, suppose that an attack would be overly troublesome if it would shut down the given DSM market for 12 hours. In each 12 block period, the attacker needs to pay the spread on the total threshold amount, totalling 10k (in units of Gyroscope stablecoins, or GYD). To last over 12 hours, they would need to repeat this 300 times, which results in a total cost of 3m GYD. This is fairly sizable considering that this just shuts down one DSM market but that other DSM markets may remain operational. If the attacker wants to shut down all DSM markets, then the costs would scale by the number of assets and be even more expensive.

Implementation. The flow rate circuit breaker is implemented in `vaultSafetyMode.sol`. In this contract, the function `_updateVaultFLowSafety` computes updated flows for an individual reserve asset and checks if the flow in the current transaction should be allowed and whether safety mode should be activated. It does this by checking the computed

flow measure against the relevant asset's `shortFlowThreshold` (i.e., $\theta_{x,\text{flow}}$) and against `shortFlowThreshold * THRESHOLD_BUFFER` (i.e., $\varepsilon_{\text{flow}}\theta_{x,\text{flow}}$). The `shortFlowThreshold` parameters need to be set for each asset while `THRESHOLD_BUFFER` is set globally across all assets. The function `_flowSafetyStateUpdater` performs the update calculations on all assets involved in the transaction and performs state updates as indicated by the circuit breaker.

The circuit breaker parameters can be calibrated by taking into account the manipulation costs of the price checks in the consolidation mechanism. In particular, the manipulation costs of these price checks should ideally be more expensive than potential manipulation exploits of the DSM, which are bounded by the circuit breaker parameters.

4.3 Oracle guardian mechanism

A DAO-elected whitelist of oracle guardians can initiate safety mode for a defined period of time. The DAO is able to replace the list of guardians at any time. This mechanism is designed as an emergency response lever, which only has pause control, and is intended to be activated in the event of an oracle failure or if a smart contract bug is found.

Should an issue arise, ideally an oracle guardian is able to initiate safety mode before an exploit occurs. However, if an exploit occurs before an oracle guardian responds, then the excessive flow rate circuit breaker will kick in first to trigger safety mode, after which the oracle guardians can extend the safety mode time length.

The oracle guardian mechanism is implemented in `vaultSafetyMode.sol`. The function `activateOracleGuardian` can only be called by whitelisted guardians and trigger safety mode for up to `safetyBlocksGaurdian` number of blocks for minting or redeeming of specified reserve assets.

5 Pricing LP Shares

In order to use LP shares in asset pools as collateral in a protocol, there need to be resilient ways to price the value of LP shares. Such methods need to be robust to manipulation of the state of the asset pools. For example, AMM pools provide algorithmic price quotes depending on the asset composition of the pool, and thus anyone can trade in the pool to manipulate the ratio of assets in the pool. For this reason, it is not a robust pricing method to simply add the values of all assets in the pool as this as this can be manipulated within a transaction.

Resilient pricing methods need to transform the portfolio value calculation of a pool into an equivalent computation that only involves quantities that are difficult to manipulate. For AMM pools, this means that the transformed computation should not involve balances of assets in the pools but instead should involve invariants about the pool structure and reliable oracle prices for underlying assets. Resilient pricing is an issue for more than just AMM pools. Other quantities about pools can also be manipulated, such as the overall level of assets in the pool and the supply of LP share tokens, considering that shares can be minted or redeemed during a transaction. For example, a 'donation attack' occurred in Cream that exploited the malleability of pool values in simpler investment vault tokens [9]. Similar exploits are possible in some AMM pool implementations as well as more general contracts that wrap underlying assets.

In this section, we will first cover pricing methods specific for AMM LP shares that are resilient to asset balance manipulations and then discuss methods that make LP share pricing resilient to donation attacks.

5.1 Constant product pools

For a given constant product pool (e.g., Uniswap v2 or Balancer weighted pools) containing assets $1, \ldots, n$, define the following:

$$\begin{aligned} w_i &= \text{weight of asset } i \\ r_i &= \text{balance (in \# tokens) of asset } i \\ p_i &= \text{price of asset } i \text{ provided by a reliable oracle} \\ S &= \text{total \# LP shares} \end{aligned}$$

Note that for convenience, S and p_i are defined differently in this section than in previous sections.

The invariant of the constant product pool is

$$L = \prod_{i=1}^{n} r_i^{w_i}$$

Note that the token balances are easily manipulable since the pool allows one token to be traded for another, but the product L is not. We can also presume that the prices p_i are not easily manipulable (e.g., the oracle prices went through the methods developed in the previous sections).

To calculate LP share pricing in a manipulation-resistant way, we will need to express the pricing of LP share tokens in terms of the manipulation-resistant variables $w_i, p_i, L/S$. We will discuss separately later when L/S is manipulation resistant, and whether we need additional protections for this quantity.

The portfolio value of the AMM pool can be calculated as

$$\text{Pool value} = L \prod_{i=1}^{n} \left(\frac{p_i}{w_i} \right)^{w_i}$$

Then, in turn, the LP share price can be calculated as

$$p_{\text{LP share}} = \frac{\text{Pool value}}{S} = \frac{L}{S} \prod_{i=1}^{n} \left(\frac{p_i}{w_i} \right)^{w_i}$$

5.2 2-CLPs

This section is intentionally blank and will be added later.

5.3 3-CLPs

This section is intentionally blank and will be added later.

5.4 E-CLPs

This section is intentionally blank and will be added later.

5.5 Manipulation-resistance of L/S

Suppose that most supply of a pool token is held in a lending market and can be borrowed. Suppose an attacker borrows all of the pool tokens from the lending market in a layered way (e.g., by borrowing some pool tokens against collateral, depositing these pool tokens as collateral and borrowing against these tokens in waves). The state of the lending protocol at this point is that it has a lot of registered deposits of the pool tokens used as collateral but holds no (or little) pool tokens in reserves. The attacker can then redeem the pool tokens they hold from the pool itself, which would bring L and S close to zero (but L/S would remain the same thus far). If the attacker then 'donates' a small amount of underlying tokens to the pooly by sending to the pool contract address, they can make L look bigger but S remain the same. For instance, if only a small amount is left in the pool after the attacker's redeem, the attacker need only donate the same small amount to double L, thus doubling L/S In this state, the lending protocol would think that L/S has increased and so would calculate the price of a pool token to be an inflated amount. The attacker can then max out their borrow position against the inflated value of pool tokens and then unwind all of their positions to leave the protocol with bad debt. The result is a donation attack, such as in [9], noting that all of this can be achieved in a single transaction with a flash loan.

For LP share pricing to be resilient to donation attacks, there need to be guardrails on how L/S is computed. One solution is to use more robust accounting methods for pool balances, e.g., so that the only way to add to pool balances is through adding liquidity as opposed to sending tokens to the contract. The current Gyroscope CLP implementations builds on Balancer v2, which uses robust accounting methods and so disallows donation attacks. Thus for Balancer pools, we have the result directly that L/S is hard to manipulate in this manner. Other pool implementations may require extra guardrails in the LP share pricing methods to confirm the value of L/S is accurate. Another option is to store a measure of past L/S values for a pool for reference as a sanity check. In normal operation, the L/S of a pool will only increase from the pool yield compounding within the pool, and so will be expected to change slowly.

6 Conclusion

In summary, the CPF includes a consolidation mechanism that contains is a sequence of price checks that cross-reference information from various sources, a sequence of circuit breakers that pause the system when state changes trigger large uncertainty about oracle security, and a library of LP share pricing formulations that are resistant to manipulation of inputs.

Together, these largely achieve the design aims laid out in Section 1. Supposing the price checks in Section 3 and the circuit breakers in Section 4 pass, Chainlink prices are used as oracle outputs, which allows the speed desired in Aim 1. CPF parameters can be calibrated so that checks are expected to pass when there aren't large manipulations, which means the CPF will be expected to remain live except in niche cases when data integrity is less certain

and price checks and circuit breakers trigger pauses (Aim 2). These same price checks and circuit breakers have quantifiable manipulation costs, meaning that even if a oracle prices were corrupted, an attacker would face high costs to manipulate other things on-chain so that the price checks and circuit breakers don't trigger (Aim 3). Similarly, should Chainlink prices become stale, this is likely to be detected (Aim 4) by triggering price checks to fail. Even if the underlying blockchain were to stop for a period of time so that prices may be stale on chain restart, the price check against exchange-signed prices also checks the timestamp, and an ETH price decline would likely trigger this price check to fail until after Chainlink prices updated. DoS attacks to trigger price checks or circuit breakers to fail also requires costly on-chain manipulation (Aim 5). Resilient LP share pricing methods are developed in Section 5 so that wrapped tokens and LP shares can be priced on-chain without adding manipulation vectors (Aim 6).

References

[1] Bentley, M. Manipulating uniswap v3 TWAP oracles. https://github.com/euler-xyz/uni-v3-twap-manipulation, Nov. 13, 2021.

[2] bertcmiller. Tweet. https://twitter.com/bertcmiller/status/1510249220967739398?t=Cf2PvmdsWyraKHNqOzYhwQ&s=19, 2 April 2022.

[3] Breidenbach, L., Cachin, C., Chan, B., Coventry, A., Ellis, S., Juels, A., Koushanfar, F., Miller, A., Magauran, B., Moroz, D., et al. Chainlink 2.0: Next steps in the evolution of decentralized oracle networks. *Chainlink Labs* (2021).

[4] Coinbase. Get signed prices. https://docs.cloud.coinbase.com/exchange/reference/exchangerestapi_getcoinbasepriceoracle-1, 15 Sep 2022.

[5] Compound. Open price feed. https://docs.compound.finance/v2/prices/, 15 Sep 2022.

[6] Ellis, S., Juels, A., and Nazarov, S. Chainlink: A decentralized oracle network. https://link.smartcontract.com/whitepaper, Sep. 4, 2017.

[7] OKX. Get oracle. https://www.okx.com/docs/en/#rest-api-market-data-get-oracle, 15 Sep 2022.

[8] Yang, Z., Klages-Mundt, A., and Gudgeon, L. Oracle counterpoint: Relationships between on-chain and off-chain market data. https://github.com/tamamatammy/oracle-counterpoint, 18 May 2022.

[9] yearn. Incident disclosure 2021-10-27. https://github.com/yearn/yearn-security/blob/master/disclosures/2021-10-27.md, Oct. 27, 2021.

should not be relied upon for accounting, legal or tax advice or investment recommendations. This paper may contain experimental code and technical designs that may not be ready for general use. This paper reflects the current opinions of the authors and is not made on behalf of Superluminal Labs or its affiliates and does not necessarily reflect the opinions of Superluminal Labs, its affiliates or individuals associated with Superluminal Labs. The opinions reflected herein are subject to change without being updated.