PRINT3R: dvAMM-based Permissionless Markets

Whitepaper v.0.1

18th April, 2024

Abstract

This paper presents the Print3r protocol, an adaptable smart contract system for trading a near-infinite array of perpetuals markets, addressing some of the predominant challenges faced by existing perpetual protocols, while introducing novel concepts and solutions capable of elevating the model to the next level.

In this paper, we introduce our unique approaches to simultaneously solving major problems that exist with decentralized perpetual protocols today, such as:

- Scalability
- Lack of decentralization
- Illiquidity in nascent markets
- Permissioned platform maintenance
- Permissioned introduction of new perpetual markets

At the core of our system, we introduce three novel concepts:

1. *dvAMM Model*: A new dvAMM model that enables dynamic rearrangement of a central liquidity pool among numerous pairs.

2. *Permissionless Creation of Perpetual Markets*: A new system enabling users to create new perpetual futures markets for virtually any token, as easily as creating a new liquidity pool on Uniswap.

3. *Decentralized Keeper Network*: A fully decentralized keeper system that actively incentivizes / rewards participants for perpetually maintaining the integrity of the protocol, enabling the scaling of markets to a near-infinite capacity, and punishing bad actors attempting to harm the system.

The Print3r protocol aims to introduce a meritocratic system, where incentives are heavily-aligned with the perpetual expansion and scaling of liquid, decentralized and efficient perpetual futures markets for any, and every crypto-asset imaginable.

Contents

1. Introduction

1.1 Motivation

Our motivation for decentralized perpetuals are deeply aligned with the principles of fully-decentralized and scalable systems like Uniswap.

Uniswap essentially allows users to swap from one token to a higher-conviction token, betting that the price will increase over time. However, users are strictly limited to holding that asset in spot.

With a fully permissionless, decentralized perpetual futures protocol acting as a supplementary layer to Uniswap, we can expand user optionality from simple, uni-directional speculation to an omni-directional form of speculation. Users can place high-conviction bets on whether the price of any asset will go up or down, enabling them to more flexibly speculate on the prices of crypto-assets.

This ability can be unlocked instantly, without a project having to struggle for years to generate enough traction to potentially get listed as a perpetual futures market on a centralized exchange.

The introduction of permissionless markets enables all builders instant access to leveraged trading on their digital assets.

This potential isn't limited to ERC-20 tokens but also extends to other digital assets, including, but not limited to:

- ERC-721 PFP collections
- ERC-1155 GameFi assets
- ERC-404 fractionalized NFTs

1.2 Choice of Architecture

The most prevalent systems for decentralized perpetual ecosystems generally fall under the umbrella of either an AMM or an OrderBook based model.

We selected the AMM model as it aligns with our idealistic view of the role Decentralized Finance will play in the broader financial ecosystem.

We believe that the core goal of decentralized finance is to empower individuals with access to the creation and usage of financial tools that would otherwise be inaccessible in traditional financial systems. Essentially, users should be able to "Be their own Bank."

For this reason, we are of the opinion that complexity should be avoided at all costs, as it raises the barrier to entry, preventing everyday users from actively participating in and benefiting from financial systems - a common problem that has plagued traditional finance for years.

As a result of this core belief, we believe that the AMM model far more effectively tackles this problem, enabling liquidity providers to deploy capital in a single click and avoid the complexity associated with becoming a liquidity provider on an orderbook, an action typically reserved for professional market makers.

This simplicity for the deployment of liquidity can create a positive feedback loop, where markets become more liquid, facilitating trading in higher volumes, which generates a greater amount of fees for the liquidity provider, attracting more liquidity, and so on.

If mass adoption is truly the goal, we believe that AMMs provide a more efficient medium to accomplish this than a traditional order book model.

1.3 Protocol Features

Below is a high-level overview of all the features introduced in this smart contract system:

1. ***Permissionless Markets***: The core feature of the Print3r protocol is the ability to create permissionless perpetual futures markets for virtually any asset.

   Upon launching, using the Print3r protocol, projects are able to instantly spin up a perpetual futures market to enable leveraged trading (both long and short) of their token, or asset.

   The project will receive 10% of all trading fees generated from the pool, thus incentivizing them to encourage, and actively promote the pool's usage.

2. ***dvAMM***: Pools deployed through the Print3r protocol can be deployed as either vAMMs (virtual AMMs), or dvAMMs (dynamic virtual AMMs).

   Virtual AMMs simply provide virtual exposure to an asset that may differ from the underlying liquidity.

   Dynamic virtual AMMs are a new primitive that enable a liquidity pool to provide virtual exposure to a large number of assets, all of which can differ from the underlying liquidity.

dvAMMs enable dynamic reallocation of liquidity between assets, so that pool configurators can optimize a pool based on demand to maximize utilization percentage for each asset.

This also solves the issue of liquidity being overly fragmented across separate, single-asset AMM pools.

Our main vision for this model is to support the creation of indexes, enabling liquidity providers to selectively provide liquidity to specific asset baskets, such as blue-chip coins, NFTs, or a Memecoin 50 index.

This model can reduce / simplify consumer choice between an infinite number of liquidity pools, and enables LPs to selectively allocate liquidity to the category of assets they want exposure to.

3. ***Reproducible and Scalable Algorithms***: One of our major design choices has been to ensure that markets require as little configuration / maintenance as possible, making it feasible for the average user to create and maintain a market.

   A protocol like Uniswap encapsulated this perfectly with their V2, using the beautifully simple x*y=k formula. [1]

   The Print3r protocol implements this by introducing formulas that can be universally applied to any market.

   Our formulas for funding rates, borrowing rates, price impact, and fee structure are all designed from first principles to incentivize balanced markets, not only between traders (long and short) but between liquidity providers too (liquidity backing long positions, and liquidity backing short positions). We believe that this approach is more scalable over a long time horizon.

4. ***Reduced Risk for Liquidity Providers***: The Print3r Protocol greatly reduces the risk exposure for liquidity providers compared to prior systems.

   As profitable positions are settled through the liquidity pool, we introduce heavy incentives for balanced markets within our system to ensure that longs and shorts offset each other's gains and losses.

   We have introduced Auto Deleveraging (ADLs) to minimize the risk of insolvency and act as an added protective feature for liquidity providers.

   Once the Pnl to Pool ratio becomes excessive, any market participant can receive a percentage-based reward for ADLing highly-profitable positions as an incentive for maintaining market solvency.

5. ***Advanced Order Types***: The Print3r protocol allows users to create simultaneous stop loss (SL) and take profit (TP) orders alongside a new position.

   SL / TP orders are now tied to positions to ensure they're cleared once the position is closed. Traders can express more flexibility over these orders, easily adjusting and removing them.

6. ***Referral System***: Our referral system enables users who refer other users to the platform to earn rebates for all of the volume generated under their referral code.

   Users who use a referral code receive a fee discount, depending on the tier of the code. Discounted fees are rebated back to the referrer, in real-time.

7. ***Decentralized Keeper Network***: The Print3r protocol is the first to achieve a fully-decentralized keeper network, enabling any user to participate in securing the ecosystem, while simultaneously being fiscally rewarded for doing so.

   This financial incentive structure empowers users with another avenue to benefit from by participating in the ecosystem, while strengthening the network and increasing the speed, throughput, scalability and security of the protocol proportionally.

8. ***Gas Rebates***: For keepers to operate at net 0 cost, the end-user is responsible for providing their gas for execution.

   Our gas rebate system ensures that users only pay the minimal required gas fee, with any excessive execution costs being rebated to the user, masking the cost as a simple, inexpensive gas fee.

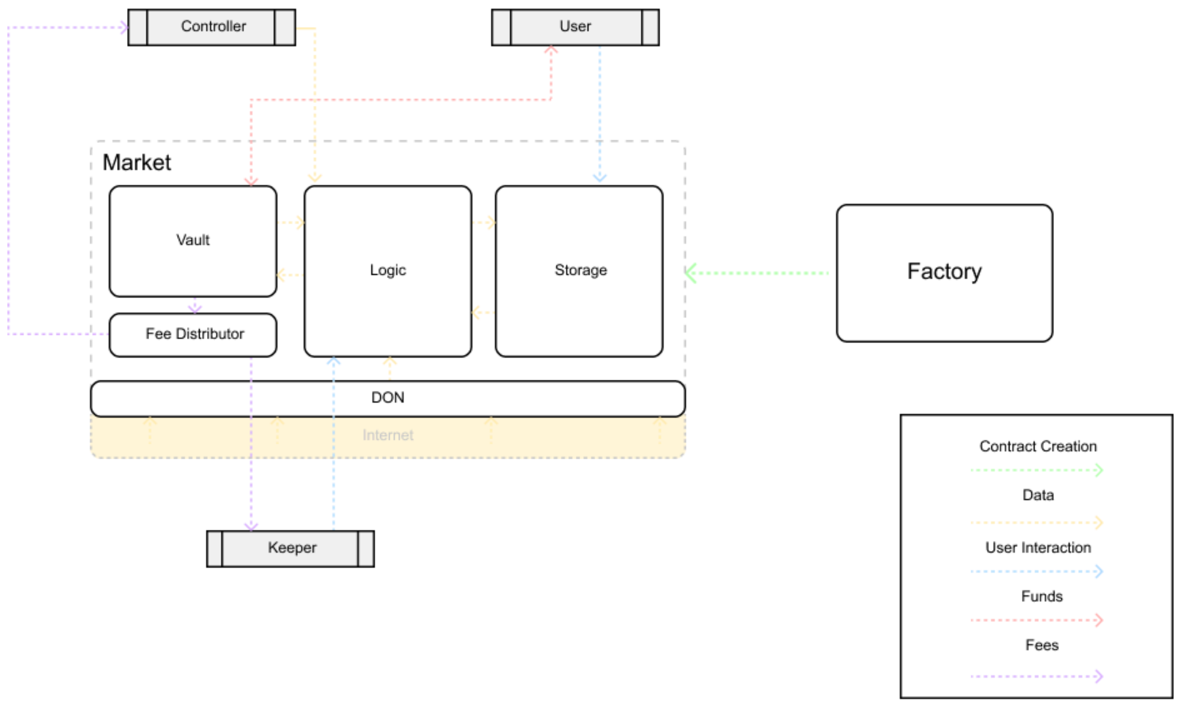
9. ***Revenue Sharing***: To incentivize the creation of pools, pool creators will receive a 10% cut of all fees generated by their pool.

   This directly incentivizes users to actively attract new users to interact with their pool and opens up a new avenue of opportunity for protocols.

10. ***SDK Integration***: We plan to develop an SDK to widget-ize interaction with our ecosystem, allowing users or projects to host their own customizable perpetual futures pool natively on their websites.

2. Protocol Details

2.1 High Level Overview



The system begins with a factory contract, which is responsible for the creation of new markets based on a set of input parameters configured by the user responsible for the market's creation (Controller).

Controller Responsibilities

The controller is responsible for configuring the market's parameters, which impact the logical component of the market. They are responsible for:

• Fee Distribution: The controller can set the intensity by which fees will be charged along a scale. They can opt to minimize fees to skew the benefit towards traders, or maximize fees to skew the benefit towards liquidity providers.

Controllers can also determine the frequency upon which fees are withdrawn and distributed to the necessary parties, although they will have no direct control over the fee amounts, or where the fees are distributed to.

• Market Configuration: Controllers are responsible for setting the caps for parameters like position leverage, funding velocity, and price impact intensity.

To incentivize the controller to maintain their pool and configure it favorably for its users, they receive a total of 10% of all fees generated by the market.

As pools will operate in a free-market environment, the pools that are the most favorably configured and maintained stand to attract the largest number of users.

Market Operations and MEV Prevention

The majority of market operations are subject to a two-step process to prevent issues arising from MEV (maximal extractable value) and the risk of malicious users attempting to manipulate the system.

1. *Request*: The user places a request, serving as an intent of an action they want to take within the market. This consists of deposits, withdrawals, and perpetual positions.

   All user actions are subject to a fee, which is accumulated within the market to incentivize active participation from liquidity providers, the keeper network, and pool controllers.

2. *Fulfillment*: Once a request is created and stored in storage, it is picked up by a keeper who first validates it, then fulfills it if the intent is valid.

   Invalid intents will simply unroll any state changes and refund the user's injected capital, if any. Valid intents will update the state of the market's storage, and any funds will be stored within the vault.

   The keeper who fulfills the request will be rewarded with a percentage of the action's fee.

Keeper System

Any user can (and is incentivized to) actively run their own keeper node.

A keeper is essentially an autonomous agent responsible for validating and executing transactions within the system. The more keepers that exist within a system, the higher the throughput and the faster the speed of transaction execution.

In a free market, keepers must compete with each other to execute transactions faster than their counterparts, meaning that over time, the maximum speed for transaction execution should theoretically be reached for each market.

Keepers are responsible for the following:

- Executing Trades
- Signing Prices
- ADLs
- Liquidations
- Executing User Deposits / Withdrawals

ADL and liquidation keepers are responsible for keeping markets solvent, while the rest of the keeper network is responsible for ensuring high transaction throughput and precise, low-latency price data.

2.2 Core Components

2.2.1 Permissionless Markets

One of the biggest risks to consider in designing a system to facilitate the creation of permissionless markets, is pricing strategies.

A malicious user could create a market and manipulate pricing in their favor to extract excessive profit from the pool's liquidity providers. Or, they could spoof a pricing strategy by providing a fake Oracle, then updating it to send invalid price data to the system.

For our solution to this problem, we host a custom pricing system on Chainlink’s decentralized oracle network (DON).

This essentially allows us to connect from the Blockchain, to the internet, to make API calls to any trusted and high-quality data providers, aggregate the retrieved data from multiple sources, then automatically serve it on-chain.

This solution provides the following benefits:

1. Pricing data is highly-accurate, and low-latency, being aggregated from high-quality price sources, like centralized exchanges and other trusted data providers.

   By aggregating data from multiple sources, we can ensure that it’s tamper proof, and any outliers are discarded.

2. The retrieval cost can be offset to users, enabling the system to operate at a net-zero cost. A small fee of just a few cents is charged to the user in wei, which is then converted into LINK to fund the subscription.

3. Pricing data is trust-less and decentralized, meaning that there's no risk that keepers, or even the team can tamper with pricing data.

4. The number of assets we are limited to is the number of assets with reliable pricing available on the internet, which has virtually no limit and is ever expanding.

Our pricing system works as follows:

1. A user creates a request for an action (Trade, Deposit, Withdrawal etc.)

2. The user pays a small fee in Ether, which is used to pay the DON subscription.

3. A price request is packaged, and sent to the DON, which then fetches and aggregates pricing data for the asset(s) from multiple sources across the internet, specific to the block at which the user created their request.

4. The DON returns a packaged request back to the smart contract after validating and ensuring the data's integrity.

5. The keeper then picks up this request, unpackages the pricing data, and uses the unpacked prices to execute the user's request.

The main trade-off with our system is our ability to expand to new Blockchains. As we utilize Chainlink, this system is only feasible on Blockchains that support the Chainlink ecosystem. At the time of writing, these chains are: Ethereum, BNB Chain, Polygon (Matic), Avalanche, Fantom, Arbitrum, Optimism and Base.

2.2.2 dvAMM

dvAMM stands for Dynamic Virtual AMM. Unlike a regular AMM, where the underlying assets deposited into a pool facilitate and limit usage to the amount of those underlying assets, a dvAMM allows the underlying assets to differ from the assets supported by the pool. This enables more abstract markets to be supported by less volatile liquidity, reducing the risk of impermanent loss taken on by liquidity providers.

We've enhanced this model to allow a single vAMM to dynamically support hundreds of assets by dynamically shifting the proportion of liquidity allocated to each individual asset.

This liquidity model provides two core benefits:

1. *Limiting consumer choice*: Making it easier for the protocol to accumulate liquidity and for users to become liquidity providers.

2. *Maximizing utilization percentage based on demand*: If an asset supported by the pool sees a surge in demand, more liquidity can be allocated to that asset to facilitate the expected increase in trading volume.

   This method allows the pool to maximize fee generation and percentage utilization of the assets in the pool without exposing users to excessive risk.

Upon creation, the pool deployer is able to select whether to deploy a dvAMM, or a regular vAMM. dvAMMs are generally better suited for facilitating trading within liquid, and less-volatile markets, so are highly suited to providing indexes for baskets of assets.

For example, a user could create a Memecoin-50 index, to facilitate trading of the 50 most popular memecoins.

Regular vAMMs are generally better suited for more volatile markets, or project owned pools.

As regular vAMMs are inherently less risky than dvAMMs, as they require less trust, front-ends are able to implement a trust-mechanism, to score pools based on a number of factors.

Users are more likely to deposit into pools with a higher trust score.

Some factors that may be considered for a trust-score based system include:

- Is it a vAMM or dvAMM? vAMMs should be allocated a higher trust score, as they're more resistant to tampering.
- Is the pool configuration malicious in any way? Pools may have excessive fees, excessive price impact, or other parameters configured in an unfavorable way.
- How volatile is the underlying asset(s)? More volatile assets should be given less trust, as they may be subject to excessive wicks, or in the worst case, rug-pulls that send price action to 0.
- How trusted is the pool's maintainer? Pools that are maintained by trusted protocols should be given a trust premium, as they have a reputation to uphold.
- How liquid / how much volume has the pool supported? Pools that are very liquid, and / or have a lot of volume mean that users are demonstrating trust in the pool, and therefore it should be allocated a higher trust score.

2.3 Liquidity Provision

Liquidity provision is one of the core tenets in our system, as liquidity providers act as counterparties to traders. For this reason, it's crucial that liquidity providers are protected as much as possible.

We've implemented several initiatives to maximize both the profitability and safety of liquidity provision on our platform:

1. *Divided Liquidity*: Liquidity is divided between liquidity to back long positions (stored in Ether/WETH) and liquidity to back short positions (stored in USDC).

   This ensures that (as markets generally move synchronously) if long positions become profitable, the long liquidity will likely increase in value proportionally, offsetting the gains accumulated by long positions.

   For short positions, if prices go down and shorts become profitable, instead of the liquidity decreasing in value proportionally, the value of the liquidity remains fairly constant, ensuring market solvency and sufficient funds to pay out potential gains.

2. *Dynamic Fee Mechanism*: To promote a balance in the amount of liquidity available for long and short positions, we have introduced a dynamic fee mechanism for deposits and withdrawals.

   Deposits/withdrawals that promote harmony between the weight of long vs. short values are subject to minimal fees, while those that hurt the harmony are subject to a greater fee, charged along a curve.

   The total fee charged for a deposit or withdrawal is calculated as follows:

   - Base Fee: A minimal fee charged to each position, calculated by multiplying the sizeDelta by the fee percentage (e.g 1 ETH * 0.0001 = 0.0001 ETH).
   - Dynamic Fee: An additional fee charged based on how the action affects the skew between long/short value. The formula is:

$$dynamicFee \ = \ sizeDelta \cdot ((\Delta negativeSkewAccrued \cdot feeScale) \ / \ \Sigma \, poolValue)$$

   The total fee charged to the position is baseFee + dynamicFee.

3. Incentives for Balanced Markets: We have introduced numerous formulas to incentivize an equal balance between long and short positions on any given market, so that the profits of one side are generally offset by the losses of the other. These incentives include:

   - Dynamic Funding Rates: Adapted from Synthetix, each market features a funding velocity defined by the formula dr/dt = c * skew, where dr/dt represents the rate of change of the funding rate over time, c is a constant factor, and skew is the difference between long and short open interest.

     The funding rate continuously adjusts based on the magnitude and duration of market imbalances, creating a strong incentive for traders to take positions that counteract the prevailing market skew.[2]

   - Price Impact: Price impact is calculated using the following formula:

$$PriceImpact = sizeDelta * \alpha((initSkew/initialTotalOi) - (updatedSkew/updatedTotalOi)) * \beta(sizeDelta / totalAvailableLiquidity)$$

     Where α = skewScalar (a dampening factor that can be applied to minimize the impact of skew) and β = liquidityScalar (a dampening factor that can be applied to minimize the impact of illiquidity). The less liquid and more skewed the market, the higher the price impact, and vice versa. Alpha and Beta can be configured on a per-market basis depending on market conditions.

4. Auto-Deleveraging (ADLs): To ensure market solvency, we've introduced ADLs. Each pool has a configurable maxPnlFactor, which represents the ratio of PNL to pool value. When the pnlFactor exceeds the maximum configured value for a pool, ADLs are triggered, forcing positions to take some profit down a deleveraging curve, starting from the most profitable positions.

2.4 Trading

Trading in PRINT3R perpetual markets offers traders access to advanced order types, like stop loss (SL) and take profit (TP) orders. These order types are crucial for risk management in trading, so we've designed our system to minimize friction from a user experience standpoint.

SL/TP orders can now be optionally and simultaneously opened alongside a position, eliminating the tedious requirement of setting them individually, as seen in some other major protocols.

A common issue in other protocols, and a pain point for users, is that as SL/TP orders effectively function as limit orders, if the position they were intended for is closed by the user without these orders executing, they aren't wiped from storage and might affect future positions the trader opens.

To circumvent this, our system perpetually ties SL/TP orders to their intended position. If the position is closed, the SL/TP orders are cleared, preventing them from impacting future positions.

2.4.1 Gas Management and Rebates

As keepers are separate autonomous entities from the user, they require gas to execute transactions. To keep the system running at net zero cost, users are required to forward the necessary gas to the keeper along with their request.

Based on the gas price at the time of the request and the computation required for the user's given action, a predicted execution fee will be applied to the position, which can be retrieved by calling Gas.estimateExecutionFee.

By keeping track of the execution fee paid by the user and subtracting the amount of gas actually consumed from the provided fee at the end of the transaction, the keeper will rebate any excess gas back to the user. This ensures that the user only pays the minimum amount of gas necessary, essentially akin to the amount of gas the user would have consumed if they had performed all the computation themselves.

To calculate the ether spent on gas for the keeper’s transaction, we follow these steps:

1. Store the initial gas at the very beginning of the transaction.

$$uint256\ initialGas\ =\ gasleft();$$

2. Subtract the remaining gas at the very end of the transaction

$$uint256\ executionCost\ =\ (initialGas\ -\ gasleft())\ *\ tx.gasprice;$$

3. Rebate the user the delta between the amount paid, and the actual execution cost

$$uint256\ feeToRefund\ =\ request.input.executionFee\ -\ executionCost;$$

2.4.2 Borrowing Fees

To make the system fairer to all users and prevent users from taking up long and short capacity without paying fees, a borrowing fee is charged on all positions as a daily percentage.

The borrowing fee is calculated depending on how close the pool is to maximum open interest capacity. The higher the demand for open interest on a pool, the higher the borrowing fee charged to active traders. Conversely, if there's plenty of open interest available on a pool, the borrowing fees charged on active positions will be negligible.

The borrowing rate calculation is simply:

$$borrowRate = borrowFeeScale \cdot (openInterest/maxOpenInterest)$$

For example, if the borrowFeeScale is set to 0.01%, that is the maximum possible borrow fee that will be charged on a given day.

Depending on the ratio of open interest compared with the maximum open interest, borrowing fees will be anywhere between 0 and the borrowFeeScale.

The units of the borrowing rate are percentage per day. So a position with $100 in size, open for 24 hours, at a 0.001% borrowing rate, will be charged $0.001 per day.

2.4.3 Auto Deleveraging

To ensure market solvency during times of extreme volatility, perpetual futures markets created through the Print3r protocol feature auto-deleveraging (ADL).

ADLs are triggered when a market becomes highly profitable, putting the pool at risk of insolvency. A default threshold is set at a 45% PnL to pool ratio. Once this threshold is reached, ADLs can be undertaken on the most profitable positions in the market to bring the PnL to pool ratio back down to a healthy level.

Users who perform the ADLs are compensated with a percentage of the total size that is de-leveraged, incentivizing them to target the positions that contribute most significantly to the PnL to pool ratio.

The formula to calculate the percentage of the position to ADL is:

$$Percentage\ to\ ADL = 1 - ((pnl\ /\ size)\ *\ e^{-excessRatio^{2}}$$

Where:

$$excessRatio = (currentPnlToPoolRatio \,/\, targetPnlToPoolRatio) - 1$$

2.5 Assets

To avoid regulatory scrutiny, we've decided to avoid supporting the trading of regulated securities and commodities and will strictly limit markets to assets within the web3 ecosystem.

However, our pricing system essentially enables us to create a market for any asset with a price. This could open up the possibility of creating prediction/pre-market tokens for users to trade the price action of an asset that is not yet live.

This also enables us to not just limit ourselves to crypto/ERC20 tokens. We plan to support everything from ERC721s, ERC1155s, and even more primitive ERCs like ERC404s.

3. Future Direction

Our entire vision for the future consists of making perpetuals more permissionless, decentralized, and scalable.

We want any user to be able to create any market, at any time, on any chain, anywhere.

And not to mention, those markets should be near-infinitely scalable, fast, liquid, secure, and 100% decentralized.

As the underlying infrastructure and tools at our disposal improve, new opportunities will inevitably open up for us to move closer to this vision.

Whether we have to build these tools ourselves or leverage new technologies like the recently released Chainlink functions, we plan to continue iterating towards this path at a rapid pace.[3]

References

1. Uniswap. “How Uniswap Works.” Uniswap Docs.
https://docs.uniswap.org/contracts/v2/concepts/protocol-overview/how-uniswap-works
2. Synthetix. "Synthetix Perps: Dynamic Funding Rates." Synthetix Blog, Apr 27, 2023.
https://blog.synthetix.io/synthetix-perps-dynamic-funding-rates/
3. Chainlink. "Chainlink Functions Documentation." Chainlink Docs.
https://docs.chain.link/chainlink-functions