探索DeFi协议预言机实施的设计空间和挑战

撰文:Adrian Chow

Jonathan Yuen和Wintersoldier贡献

摘要：

预言机（Oracle）于保障 DeFi协议的锁定价值不可或缺，DeFi 的 500 亿美元总锁仓量当中，有 330 亿由预言机保障。

然而，预言机喂价更新时本质上的时间延迟，导致最大可提取价值（MEV, Maximal Extractable Value）一个子类型的价值提取，这被称为预言机可提取价值（OEV, Oracle Extractable Value）； OEV 包括了预言机抢先交易(frontrunning)、套利(arbitrage)和低效平仓(inefficient liquidations)。

目前有越来越多的设计实施方案可防止或减轻 OEV 的负面流失，每种设计都有其独特的取舍权衡。本文讨论现有设计的选择及其权衡，以及提出了两个新构思、其价值主张、未决问题以及发展瓶颈。

### 引言

预言机（Oracle）可说是当今 DeFi 最重要的基础设施之一。它们是大多数 DeFi 协议不可或缺的部分，这些协议依靠喂价来结算衍生品合约、平仓抵押不足的持仓等。目前，预言机保障了330 亿美元的价值，占链上总锁仓量 500 亿美元的至少三分之二1。然而，对于应用程序开发人员来说，加入预言机会带来明显的设计权衡和问题，这源于抢先交易(frontrunning)、套利(arbitrage)和低效平仓(inefficient liquidations)等的价值流失。本文将这种价值流失分类为预言机可提取价值(Oracle Extractable Value, OEV)，从应用程序的角度概述了其关键问题， 并试图在行业研究的基础上，说明在DeFi 协议中安全可靠地加入预言机的关键考量。

### 预言机可提取价值 (OEV)

本节假定读者对预言机功能，以及对推送式(push-based)和拉取式(pull-based)预言机的区别有基本了解。个别预言机的喂价可能不同。有关概述、分类和定义，请参阅附录。

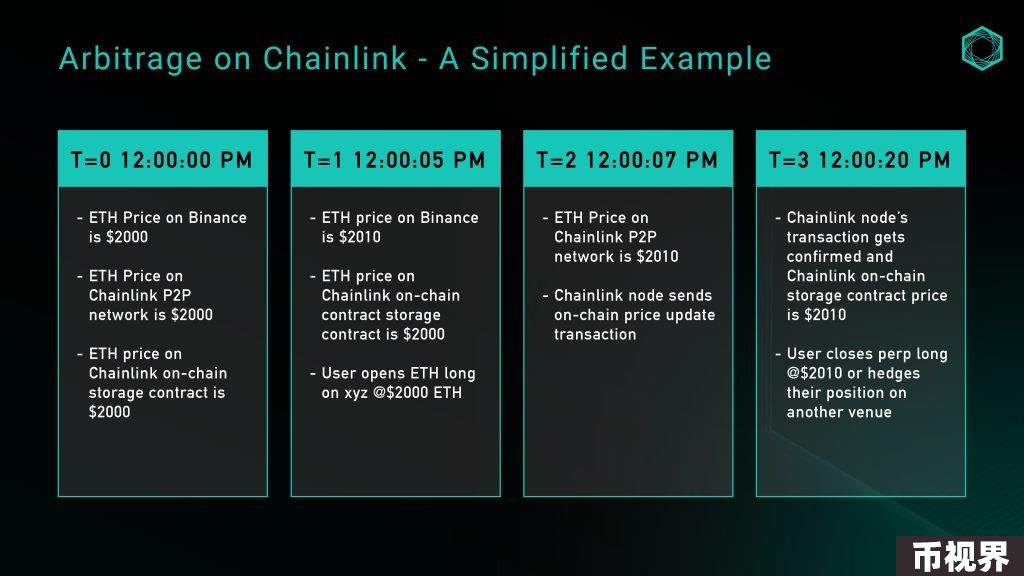
大多数使用预言机喂价的应用程序只需要读取价格：运行自己定价模式的去中心化交易所使用预言机喂价作为参考价格；为超额抵押贷款仓位存入抵押品，只需要预言机读取价格，以确定如借款价值比平仓价格等初始参数；撇除长尾资产等定价更新过于不频繁的极端情况，基本上在考虑设计系统时，预言机更新喂价的延迟并不重要。因此，预言机最重要的考量是 - 评估价格贡献者的准确性，以及预言机提供者的去中心化性能。

但如果喂价更新的延迟 是重要考虑因素，则应更为注意预言机如何与应用程序交互。通常在这种情况下，此类延迟会导致价值提取机会，即抢先交易、套利和平仓。这种MEV的子类型被称为OEV2。在讨论各种实施方案及其权衡之前，我们将概述OEV的各种形式。

### 套利

预言机抢先交易和套利在衍生品协议中被俗称为”毒流”（toxic flow ），因为这些交易是在信息不对称的情况下进行的，往往以牺性流动性提供者的成本获取无风险利润。 Synthetix 等 OG DeFi 协议自 2018 年来一直在应对这一问题，并随着时间的推移尝试了各种解决方案，以减轻这些负面外部性。

让我们以简单的例子说明；永续合约去中心化交易所xyz 在 ETH/USD 市场上使用 Chainlink 预言机，例子以ETH/USD 喂价说明 ：



虽然上面为过于简化的示例，没有考虑滑点、费用或资金等因素，但它说明了偏差阈值的角色导致价格粒度不足，从中所带来的机会。搜索者可以根据 Chainlink 的链上存储，监控现货市场价格更新的延迟，并从流动性提供者(Liquidity Provider, LP) 提取零风险价值。

### 抢先交易

抢先交易与套利类似，是另一种价值提取形式，搜索者监控内存池的预言机更新，并在其提交链上之前，抢先运行实际市场价格。这样，搜索者就有时间在预言机更新前出价交易，以有利于自己交易方向的价格成交。

GMX等这种永续合约去中心化交易所一直都是毒性抢先交易的受害者；于GMX所有预言机通过 KeeperDAO 协调协议更新前，约10%的协议利润已于抢先交易流失4。

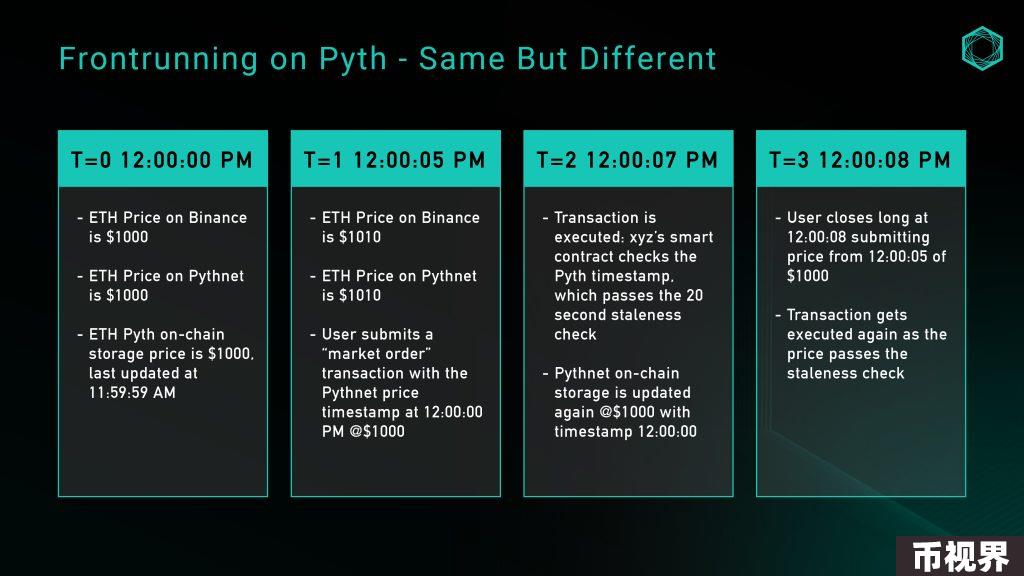
### 如果我们只采用拉取式模型？

Pyth 的价值主张之一是，使用 Solana 架构的Pythnet ，发布者可每 300 毫秒5向网络推送一次价格更新，从而维持低延迟喂价。因此，当应用程序通过 Pyth 的应用程序接口(API)查询价格时，可以检索最新价格、将其更新到目标链的链上存储、并在一次交易中执行应用程序逻辑中的任何下游操作。

如以上所述，应用程序能够直接查询 Pythnet 的最新价格更新、更新链上存储、并在一次交易中完成所有相关逻辑，这不就有效地解决了抢先交易和套利问题？

也不尽如此 - Pyth 的更新，赋予了用户选择在交易中使用哪些价格的能力，这可能会导致逆向选择（adversarial selection)（毒流的另一种修辞）。虽然链上存储价格必须随时间推移，但用户仍可选择任何满足这些限制条件的价格 - 意味着套利仍然存在，因为它允许搜索者在使用过去的价格之前看到未来的价格。 Pyth 的文档 6 建议，防范这种攻击媒介的一个简单方法是加入期效检查（staleness check），以确保价格够近期- 但是，更新交易数据于下一个区块中必须有一定的缓冲时间，我们该如何确定最佳时间阈值？

让我们以永续合约去中心化交易所 xyz 为例进行分析，而今次他们使用的是 Pyth ETH/USD 喂价，期效检查时间为 20 秒，这意味着 Pyth 价格的时间戳，必须处于执行下游交易的区块时间戳的 20 秒之内：



一个直观的想法是简单地降低期效检查阈值，但较低的阈值可能会导致无法预测区块时间的网络回复，从而影响用户体验。由于 Pyth 的喂价依赖于桥接，因此需要足够的缓冲来 a) 为虫洞守护者（Wormhole guardians）提供证明价格的时间，b) 允许目标链处理交易并将其纳入区块。下一节将详细介绍这些权衡。

### 平仓

平仓是任何涉及杠杆的协议的核心部分，而喂价更新的粒度，于决定平仓效率举足轻重。

以基于阈值的推送式预言机来说，当现货价格达到阈值但不符合预言机喂价预设的参数时，价格更新的粒度（或粒度不足）会导致错失平仓机会。这以市场低效率的形式带来了负外部效应。

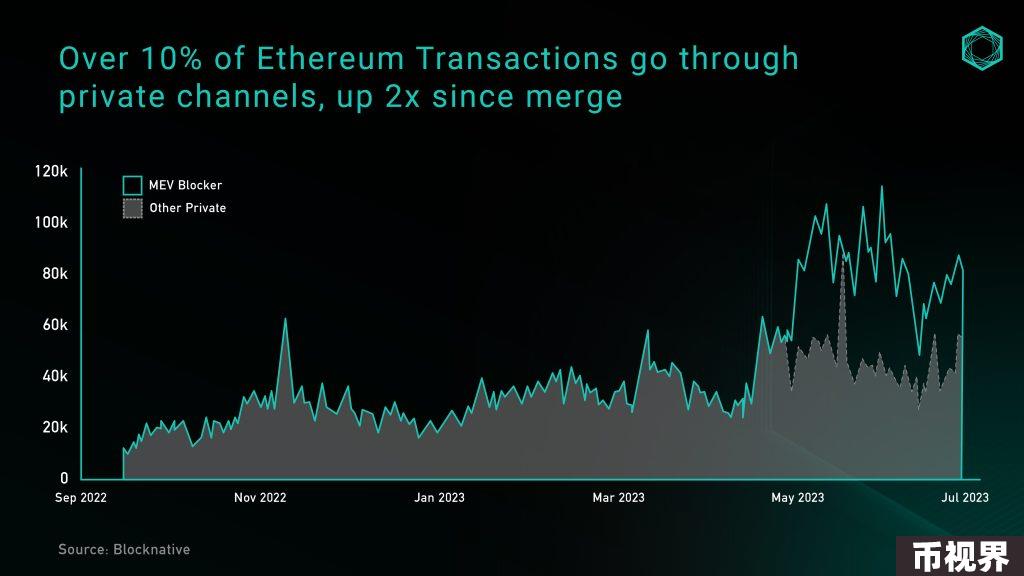
当平仓发生，应用程序通常会支付部分平仓抵押品，有时还会向发动平仓的用户提供奖励。例如，2002年Aave 仅在主网上就支付了3,790 万美元的平仓奖励7。这明显过度补偿了第三方，并且为用户带来不佳的操作 。此外，当存在可提取价值时，随之出现的 Gas Wars (Gas 竞拍行为)会导致价值从应用程序中流失，从而流向MEV供应链。

### 设计空间和考量

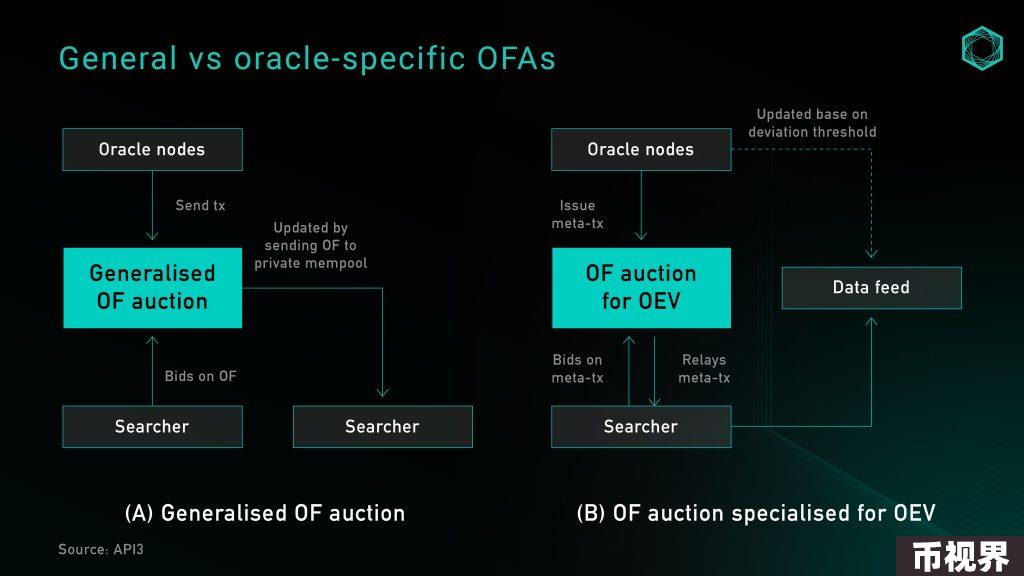
考虑到上述问题，下文将讨论基于推送式、拉取式和替代设计的各种实施方案，各自于解决上述问题的有效性及当中的取舍；取舍的形式可以是附加的中心化和信任前设，又或是不佳的用户体验。

预言机专用的订单流竞价（Order Flow Auctions，OFA）

订单流竞价OFA已成为消除MEV 产生的负外部效应一种解决方案。广义上，OFA是一种通用的第三方竞价服务，用户可以向其发送订单（交易或意图），而提取MEV的搜索者则可以竞价获得对其订单运行策略的独家权利。很大部分的竞价收益会退还给用户，以补偿他们在这些订单中创造的价值。近来OFA的采用率激增，超过10%以太坊交易都于私人渠道（私人 RPC/OFAs）进行（图 3），相信尚会进一步催化增长。



在预言机更新中，实施通用OFA的问题在于预言机无法了解基于标准规则的更新，是否会产生任何OEV，如果不会，则会在预言机向竞价中发送交易时带来额外延迟。另一方面，精简OEV，和将延迟减至最低的最简单方法是将所有预言机订单流提供予单一主导搜索者。但这显然会带来极大集中化风险，可能会助长寻租行为以及审查，并导致低用户体验。



不包含现有基于规则更新的预言机专用OFA的价格更新仍於公共内存池进行。这让预言机的价格更新，以及随之产生的任何可提取价值，都得以保留在应用层中。作为副产品，它还允许搜索者请求数据源更新，而无需预言机节点承担更频繁更新的额外成本，从而提高了数据的粒度。

预言机专用OFA是平仓的理想选择，因为它能带来更细粒度的价格更新，最大限度地将资本返还给被平仓的借款人，减少支付给平仓人的协议奖励，并在协议中保留从投标人处提取的价值，以便重新分配给用户。它们还在一定程度上 - 尽管并不完全- 解决了抢先交易和套利问题。在完全竞争（perfect competition）和首价密封投标竞价(first price sealed bid auction)流程下，竞价的结果，应是接近执行机会 8的区块空间成本、由抢先交易OEV数据馈送中，所提取的价值，以及因喂价更新的价格粒度增加，而减少所产生的套利机会。

目前，要实施预言机专用的OFA，要么需要加入第三方竞价服务（如OEV-Share），又或构建一个竞价服务作为应用程序的一部分。受 Flashbots 的启发，API3 利用OEV中继器(OEV relay) （图 5）作为于设计上执行 DoS 保护服务的 API 来进行竞价。该中继器负责收集来自预言机的元交易、整理和聚合搜索者的出价，并以无信任方式重新分配收益，而无需控制出价。当搜索者中标时，更新数据源只能依靠将出价金额转移到协议拥有的代理合约，然后代理合约会用中继器提供的签名数据更新价格源。



另外，协议也可以放弃中间人，建立自己的竞价服务，获取从OEV的所有提取价值。BBOX 就是一个即将推出的协议，它希望将竞价嵌入其平仓机制，以获取OEV，并将其返还给应用程序及其用户9。

运行中央节点或Keeper

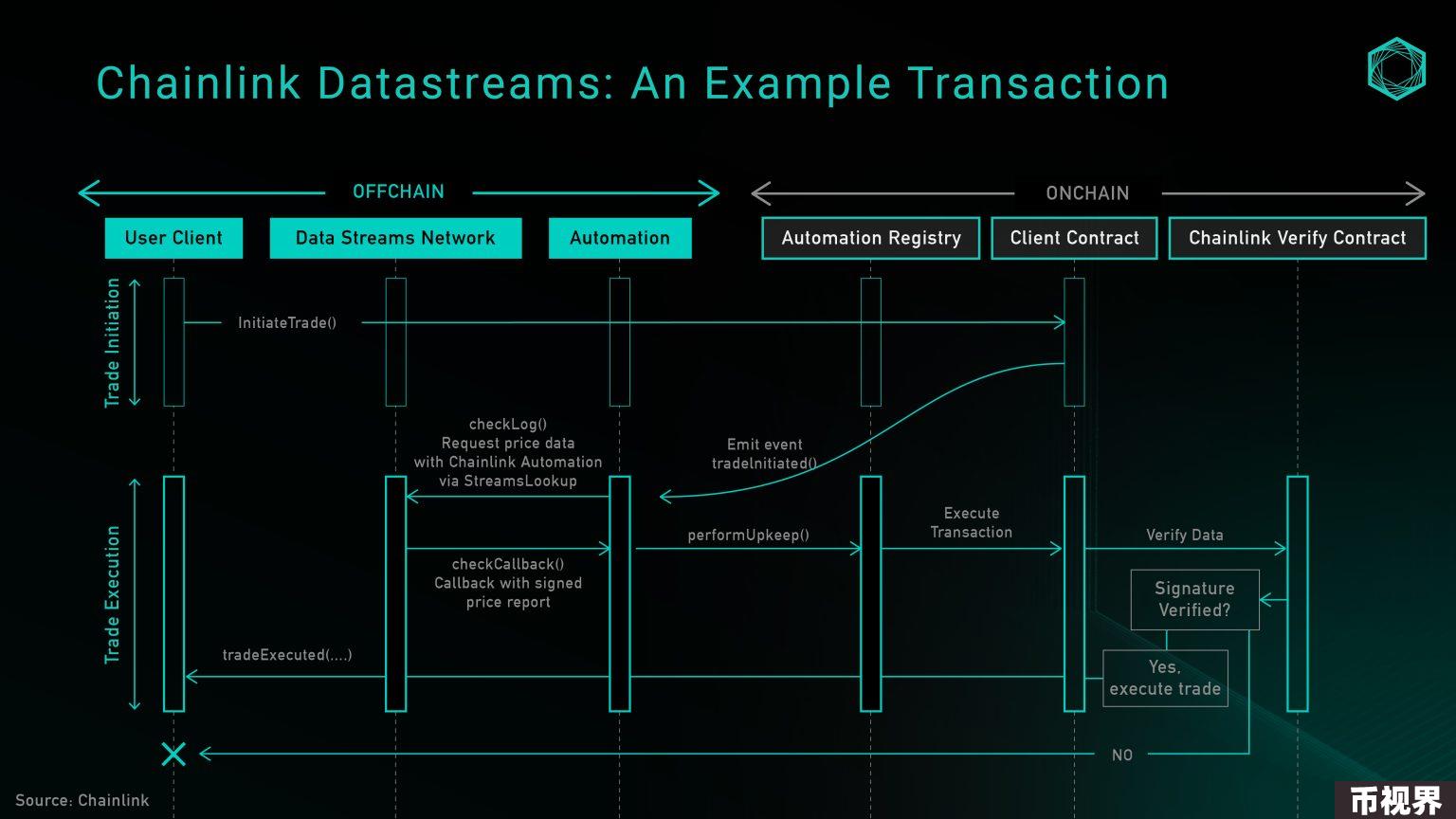
源于第一波永续合约去中心化交易所打击OEV的一个早期想法，是运行一个集中式Keeper网络(守门人网络)，聚合从第三方来源（如中心化交易所）收到的价格，然后利用类似 Chainlink 的数据馈送作为应变方案或断路器。这种模式在 GMX v110 及其后续的许多分叉中都得到了推广，其主要价值主张在于，由于Keeper网络由单一运营商运行，因此可以绝对防止抢先交易。

虽然这解决了上述许多问题，但明显地有中心化顾虑。中心化的Keeper系统，可以于未适当验证定价来源和聚合方法之下决定执行价格。在 GMX v1 的案例中，Keeper并不是一个链上或透明的机制，而是于中心化服务器上运行团队地址签署的程序。 Keeper的核心作用不仅是执行订单，而且是根据自己的预设定义 "决定 "交易价格，无法验证所使用的执行价格的真实性或来源。

### 自动化的Keeper网络和Chainlink 数据流

解决上述由单一操作员的Keeper网络所带来的中心化风险，是利用第三方服务提供商建立一个更加去中心化的自动化网络。 Chainlink Automation 就是这样一个产品，它与 Chainlink Data Streams - 即是一个全新的拉取式、低延迟预言机 - 共同提供这项服务。该产品最近刚刚发布，目前还处于闭门测试阶段，但 GMX v211 目前已经在使用该产品，它可以作为采用这种设计的系统的参考。

从高层次来看，Chainlink 数据流由三个主要部分组成：数据 DON（去中心化的预言机网络）、自动化 DON 和链上验证合约12。数据DON是一个链下数据网络，其架构类似于Pythnet维护和聚合数据的架构。自动化 DON 是由数据 DON 的相同节点操作员保护的守护者网络，用于从链上数据 DON 提取价格。最后，验证器合约用于验证链下签名是否正确。



上图展示了调用开放交易功能的交易流程，其中自动化 DON 负责从数据 DON 获取价格并更新链上存储。目前，直接查询数据 DON 的端点仅限于白名单用户，因此协议可以选择将Keeper维护工作卸载给自动化 DON(Automation DON)，或运行自己的Keeper。但随着产品开发生命周期的推移，预计这将逐步转变为无权限结构。

在安全层面上，依赖自动化 DON 的信任假设，与单独使用数据 DON 的信任假设相同，这是对单一Keeper设计的重大改进。不过，如果将喂价更新权交给自动化 DON，那么价值提取的机会就只能留给Keeper网络中的节点。这反过来又意味着，该协议将信任链克节点运营商（主要是机构）维护其社会声誉，不抢先用户进行操作，这类似于信任Lido Node节点运营商因要维护其声誉，不会因其市场份额大，而垄断区块空间

拉取式： 延迟结算

Synthetix perps v2 中最大的变化之一，是为永续合约结算13引入了 Pyth 喂价。这使得订单可以以 Chainlink 或 Pyth 价格结算，前提是它们的偏差不超过预定义的阈值，并且时间戳通过了期效检查。然而，如上所述，仅仅改用基于拉取式预言机并不能解决所有协议的OEV相关问题。要解决抢先交易，可以延迟订单的形式引入 "最后查看 "定价机制，在实践中，这将用户的市场订单分为两个部分：

交易 #1：在链上提交开立市场订单的 "意向"，并提供标准订单参数，如大小、杠杆、抵押品和滑点容忍度。同时还需支付额外的Keeper费用，用于奖励Keeper执行交易 #2。

交易 #2：Keeper接收在交易 #1 中提交的订单，要求最新的 Pyth 喂价，并在一次交易中调用 Synthetix 执行合约。合约会检查预定义的参数，如时效和滑价，如果都通过，订单就会被执行，链上价格存储会被更新，仓位将建立。 Keeper收取费用，补偿使用和维护网络的所用到的gas。

这种实施方式不会让用户有机会逆向选择在链上提交的价格，从而有效地解决了协议的抢先交易和套利机会。不过，这种设计的折衷就是用户体验：执行这个市场订单需要两个交易过程，用户需要为Keeper的操作补偿gas，同时分担更新预言机链上存储的的成本。之前是 2 sUSD的固定费用，最近则改为基于Optimism gas oracle + 溢价的动态费用，溢价将根据二层网络（layer 2）活动而变化。无论如何，这可视为牺牲交易者的用户体验以提高 LP 盈利能力的一种解决方案。

拉取式： 积极结算 (Optimistic settlement)

由于延迟订单会给用户带来额外的网络费用（和二层网络的DA 费用成比例），经过集思广益，我们再拟出了另一种订单结算模式，称之为"积极结算"，这种模式有可能降低用户的成本，同时维护去中心化以及协议的安全性。顾名思义，这种机制允许交易者以原子方式执行市场交易，系统会积极地接受所有价格，并为搜索者提供一个窗口，让他们提交证据，证明恶意下达的订单。本节概述了这构思的不同版本、我们的思考过程以及仍未解决的问题。

我们最初的想法是建立一种机制，让用户在开立市场订单时通过 parsePriceFeedUpdates 提交价格，然后允许用户或任何第三方使用喂价数据提交结算交易，并在交易确认时以该价格完成交易。结算时，两个价格之间的任何负差都将作为滑点计入用户的损益表。这种方法的优点包括减轻用户的成本负担，和降低抢先交易的风险。用户不必再负担奖励守们人的溢价，而且由于在提交订单时不知道结算价格，抢先交易的风险仍可控。不过，这仍引入了两步的结算流程，而这正是我们在 Synthetix 的延迟结算模式中发现的缺点之一。在大多数情况下，如果下单和结算期间的波动性，不超过系统界定的可盈利抢先交易阈值，那额外的结算交易可能就是多余的。

规避上述问题的另一种解决方案是，允许系统积极地接受订单，然后开放一个无权限的质疑期，在该期间可以提交证据，证明价格时间戳和区块时间戳之间的价格偏差允许进行有利可图抢先交易。

具体操作如下：

用户根据当前市场价格创建订单。然后，他们连同嵌入的 pyth 喂价字节数据传送价格﹐作为订单创建交易。

智能合约会主动验证并存储这些信息。

在链上确认订单后，会有一个质疑期，搜索者可以提交逆向选择证明。该证明将证实交易者使用了过时的喂价数据，意图在系统中套利。如果系统接受了证明，差值将作为滑点应用到交易者的执行价格中，多余的价值将作为奖励给予Keeper。

质疑期结束后，系统认为所有价格均有效。

这种模式有两个优点：减轻了用户的成本负担，用户只需在同一笔交易中为订单创建和预言机更新支付gas 费用，而不需要额外的结算交易。它还能阻止抢先交易，保护流動池的完整性，确保有一个健康的Keeper网络，有经济奖励措施向系统提交证明，证明其抢先。

然而，在将这一想法付诸实践之前，还有一些问题有待解决：

定义 "逆向选择"： 系统如何区分因网络延迟而提交过期价格的用户，以及故意套利的用户？一个初步的想法可以是，测量期效检查时段（例如 15 秒）内的波动性，如果波动性超过净执行费，该订单就会被标记为一个潜在利用。

设置适当的质疑期： 考虑到有毒订单流可能只开放很短的时间，什么是适当的时间窗口供Keeper质疑价格？批量证明可能会更符合成本效益，但鉴于订单流在一段时间内的不可预测性，很难确定批量证明的时间，以确保所有价格信息都得到证明或有充足的时间受到质疑。

对Keeper的经济奖励： 要使提交证明对受到经济激励的保存者来说是合理的，提交获胜证明的相关奖励必须大于提交证明的相关gas成本。由于订单规模不同，这一假设可能无法保证。

是否需要为关闭订单建立类似的机制？如果要的话，会怎样降低了用户体验？

确保 “不合理” 的滑点不会落到用户身上： 在闪崩情况下，订单创建和链上确认之间可能会出现非常大的价格差异。可能需要某种后备或断路器，可以考虑使用 Pyth 的 EMA 价格，以确保使用前的喂价稳定性。

中科辅助处理器 （ZK Co-processors）- 另一种形式的数据消耗

另一个值得探索的方向是使用ZK 辅助处理器，这种辅助处理器旨在获取链上状态以于链下进行复杂计算，并与此同时提供计算执行方式的证明；这种方式可无权限地验证。 Axiom 等项目使合约能够查询历史区块链数据，在链下执行计算，并提交 ZK 证明，证明计算结果是根据有效的链上数据正确计算得出的。辅助处理器开启了一种可能性，利用多个 DeFi 原生流动性来源（如 Uniswap + Curve）的历史价格构建具有操纵弹性的自定义 TWAP 预言机。

与目前只能获得最新资产价格数据的传统预言机相比，ZK 辅助处理器将扩大以安全方式提供给 dApp 的数据范围（Pyth 确实提供了 EMA 价格，供开发人员用作最新价格的参考检查）。这样，应用程序就可以引入更多与历史区块链数据协同工作的业务逻辑，以提高协议安全性或增强用户体验。

不过，ZK 辅助处理器仍处于开发初期，当中仍存在一些瓶颈，例如：

在辅助处理器环境下，大量区块链数据的获取和计算可能需要较长的证明时间

仅提供区块链数据，无法解决与非 Web3 应用程序安全通信的需求

### 无预言机解决方案 – DeFi 的未来？

解决这一问题的另一种思路是，通过从头开始设计一个基元，消除对外部喂价的需求，从而

解决DeFi对预言机依赖性。这一领域的最新发展是利用各种 AMM LP 代币作为定价手段，其核心理念是恒定函数做市商的 LP 仓位是代表两种资产预设权重的代币，并有这两种代币的自动定价公式（即 xy=k）。通过利用 LP 代币（作为抵押品、贷款基础，或在最近的使用案例中，将 v3 LP 仓位移动到不同的刻度点），该协议可以获取通常需要从预言机所获取的信息。由此，新一波趋势 - 免于所述挑战的无预言机方案都得以实现。建基于此方向的应用实例包括：

Panoptic 正构建永久、无预言机的期权协议，所利用的是 Uniswap v3 集中流动性仓位。由于当现货价格超过 LP 仓位的上限范围时，集中流动性仓位会 100%转换成基础资产，因此流动性提供者的回报与认沽期权的卖家回报非常相似。因此，期权市场的运作是流动性提供者存入 LP 资产或仓位，期权买方和卖方借入流动性并将其移入或移出范围，从而产生动态的期权回报。由于贷款是以 LP 仓位计价，因此结算时不需要预言机。

Infinity Pools 正在利用 Uniswap v3 的集中流动性仓位，建立一个无平仓、无预言机的杠杆交易平台。 Uniswap v3 的流动性提供者可以借出他们的 LP 代币，交易者存入一些抵押品，借用 LP 代币并赎回其定向交易的相关资产。赎回时的贷款将以资产基础或报价资产计价，具体取决于赎回时的价格，并可直接通过检查 Uniswap 上的 LP 组成计算，消除了对预言机的依赖。

Timeswap 正在建立一个固定期限、无平仓、无预言机的借贷平台。它是一个由贷方、借方和流动性提供者组成的三方市场。与传统借贷市场不同，它采用的是 "时间基础 " （“time-based”）的清算，而不是"价格基础"（price-based）的平仓。在去中心化交易所，流动性提供者被自动设定为总是向卖方买入，向买方卖出；而在 Timeswap 中，流动性提供者总是向借方贷款，向贷方借款，在市场中扮演类似的角色。他们还负责承担贷款违约责任，并优先获得被没收的抵押品作为补偿。

### 结论

定价数据仍然是许多去中心化应用的重要部分，而随着时间的推移，预言机所获得的总价值也在不断增加，进一步肯定其产品与市场的契合度(p产品市场契合度）。本文旨在让读者得悉并概述我们目前面临的OEV相关挑战，以及基于推送式、拉取式和使用 AMM 流动性提供者或链下辅助处理器的其他设计，其实施方案中的设计空间。

我们很高兴看到充满活力的开发者期望解决这些棘手的设计难题。如果您也在这领域开展颠覆性的项目，我们很乐意听取您的意见！

参考文献和致谢

感谢 Jonathan Yuen 和 Wintersoldier 的贡献和对谈，为本文贡献良多。

感谢 Erik Lie、Richard Yuen（Hailstone）、Marc、Mario Bernardi、Anirudh Suresh（Pyth）、Ugur Mersin（API3 DAO）和 Mimi（Timeswap）的宝贵意见、反馈和审查。

https://defillama.com/oracles (14 Nov) ↩︎

OEV Litepaper https://drive.google.com/file/d/1wuSWSI8WY9ChChu2hvRgByJSyQlv\_8SO/edit↩︎

Frontrunning on Synthetix: A History by Kain Warwick https://blog.synthetix.io/frontrunning-synthetix-a-history/

https://snapshot.org/#/rook.eth/proposal/0x523ea386c3e42c71e18e1f4a143533201083655dc04e6f1a99f1f0b340523c58 ↩︎

https://docs.pyth.network/documentation/pythnet-price-feeds/on-demand ↩︎

https://docs.pyth.network/documentation/solana-price-feeds/best-practices#latency ↩︎

Aave liquidation figures https://dune.com/queries/3247324

https://drive.google.com/file/d/1wuSWSI8WY9ChChu2hvRgByJSyQlv\_8SO/edit

https://twitter.com/bboexchange/status/1726801832784318563

https://gmx-io.notion.site/gmx-io/GMX-Technical-Overview-47fc5ed832e243afb9e97e8a4a036353

https://gmxio.substack.com/p/gmx-v2-powered-by-chainlink-data ↩︎

https://docs.chain.link/data-streams ↩︎

https://sips.synthetix.io/sips/sip-281/

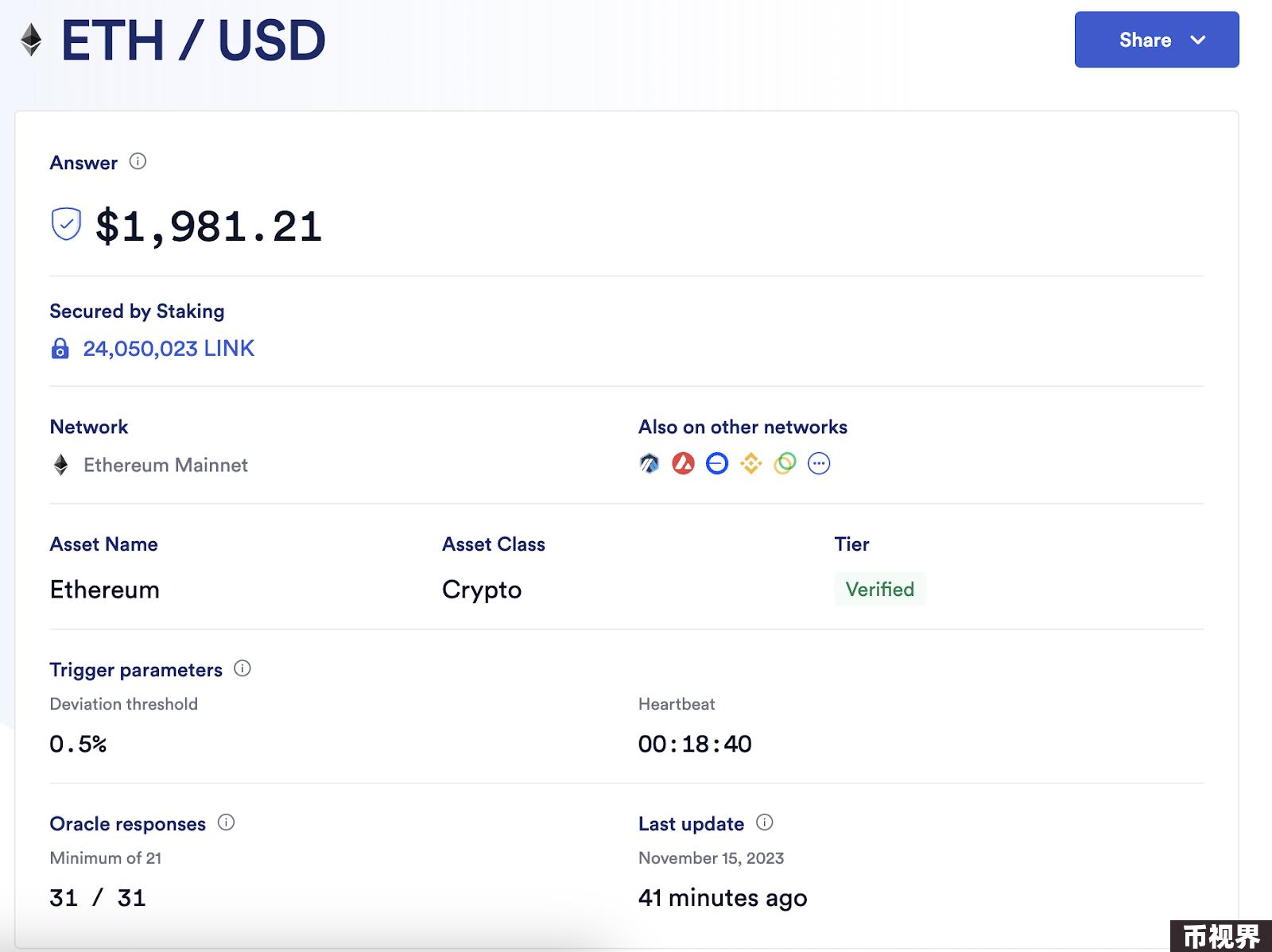
↩︎

︎︎

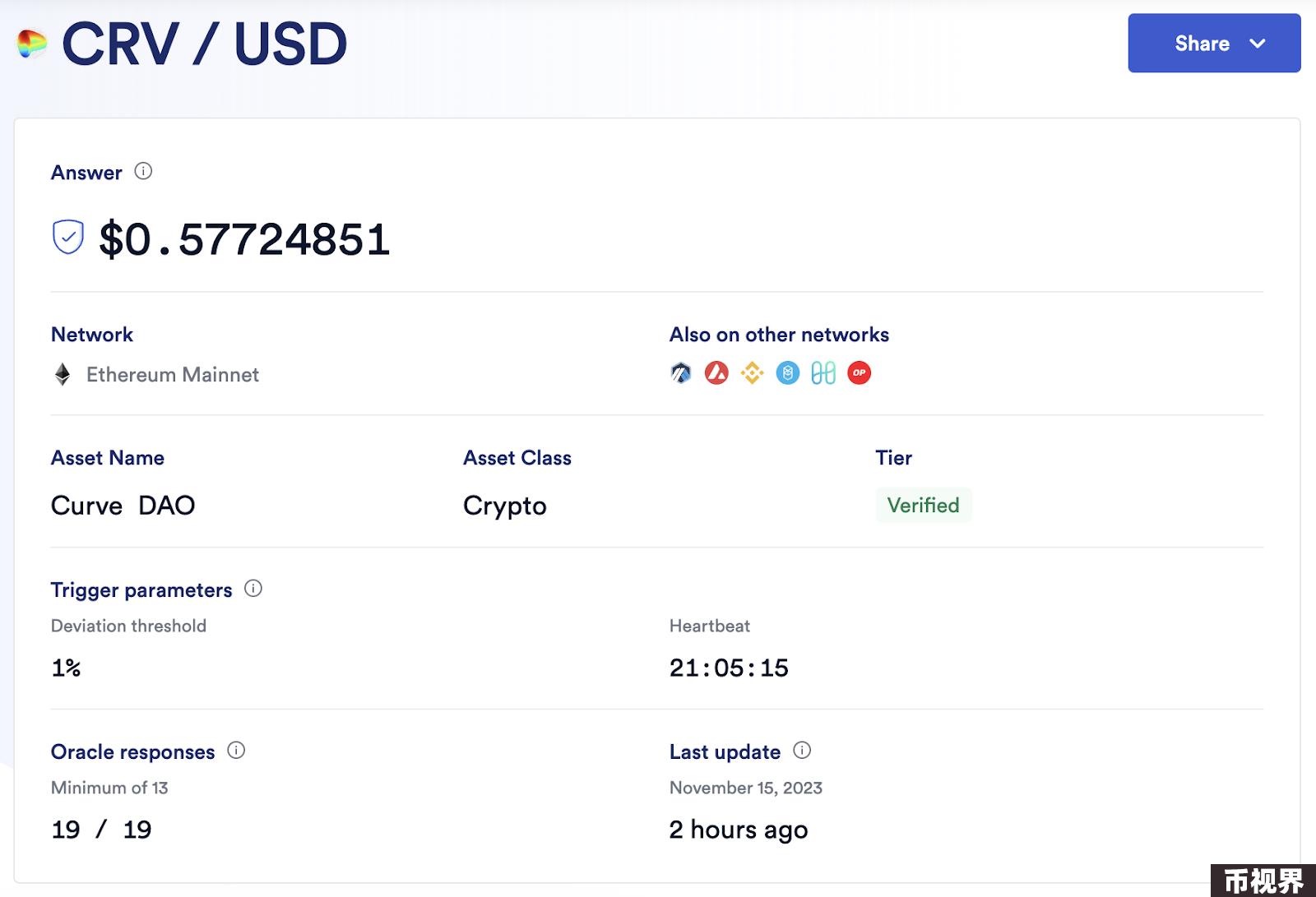
附录

定义： 推送式与拉取式预言机

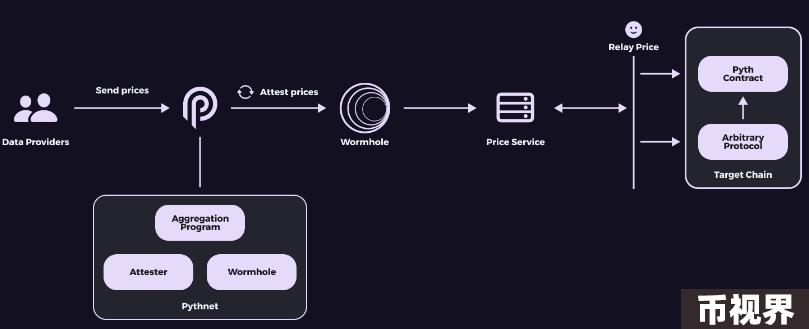
推送式预言机器于在 P2P 网络中维持链下价格，以及维持根据预先定义的链上节点更新价格。以 Chainlink 为例，价格更新基于两个触发参数：偏离阈值（偏差阈值）和心跳（心跳）。只要链下价格偏离最新链上价格 0.5%，或者心跳计时器 1 小时计时为零，下面的以太坊 ETH/USD 价格源就会更新。



在这种情况下，预言机运营商必须为每次价格更新支付交易费用，也是于成本和可扩展性之间的取舍。增加价格源的数量，支持额外的区块链，或者增加更频繁的更新，都会产生额外的交易成本。因此，具有更高触发参数的长尾资产，无可避免地具有可靠度低的价格源。下面以CRV/USD为例说明这一点- 为了使新的价格能够在链上更新，需要1%的偏离阈值，心跳为24小时，这意味着如果价格在24小时内未偏离超过1％，那么每24小时只会有一个新的价格更新。直观而言，长尾资产的价格源缺乏细致度，将不可避免地导致应用程序在为这些资产创建市场时需要考虑额外的风险因素，这解释了为什么绝大多数DeFi活动仍围绕着流动性最强、最大市值的代币而发生。



相比之下，拉取式预言机允许按需求将价格拉到链上。Pyth 是当今最突出的例子，它在链下传输价格更新，对每次更新进行签名，以便任何人都能验证其真实性，并在Pythnet上维护聚合价格，Pythnet是基于Solana代码的私有区块链。当需要更新时，数据通过Wormhole传输，在Pythnet上进行验证后，就可以无需权限地拉取到链上。



上图描述了Pyth 喂价的架构： 当需要更新链上价格时，用户可以通过Pyth API 请求更新，Pythnet 上经过验证的价格会被发送到Wormhole 合约，Wormhole 合约会观察并创建和发送一个处名的VAA，该VAA 可以在任何部署了Pyth 合约的区块链上进行验证。