# SoK: Blockchain Oracles Between Theory and Practice

Colin Finkbeiner
*University of Connecticut*
*colin.finkbeiner@uconn.edu*

Ghada Almashaqbeh
*University of Connecticut*
*ghada@uconn.edu*

*Abstract*—**Smart contract-based decentralized applications (dApps) have become an ever-growing way to facilitate complex on-chain operations. Oracle services strengthened this trend by enabling dApps to access real-world data and respond to events happening outside the blockchain ecosystem. A large number of academic and industrial oracle solutions have emerged, capturing various designs, capabilities, and security assumptions/guarantees. This rapid development makes it challenging to comprehend the landscape of oracles, understand their trade-offs, and build on them.**

**To address these challenges, we develop a systematization of knowledge for blockchain oracle services. To the best of our knowledge, our work is the first to provide extensive study of oracles while empirically investigating their capabilities in practice. After examining the general design framework of oracles, we develop a multi-dimensional systematization framework assessing existing solutions based on their capabilities, trust and security assumption/guarantees, and their underlying design architecture. To further aid in this assessment, we conduct a number of empirical experiments to examine oracle deployed in practice, thus offering additional insights about their deployment maturity, usage popularity, performance, and ease-of-use. We go on to distill a number of insights and gaps, thus providing a guide for practitioners (on the use of these oracles) and researchers (by highlighting gaps and open problems).**

## 1. Introduction

Smart contract-enabled blockchains, and their decentralized applications (dApps), are considered the core engine of Web 3.0. They facilitate complex on-chain operations, thus fostering a large variety of applications such as digital identity [1], [2], gaming [3], [4], and decentralized finance (DeFi) [5]. However, dApps are limited to actions or information within the blockchain ecosystem. Thus, access to real-world data (e.g., stock prices) and events (e.g., sport results), as well as expensive computation, is not directly available. For many applications, including derivatives [6], [7], lending protocols [8], [9], and supply chain management [10], [11], interaction with the real world in a timely manner is critical to their secure and effective operation.

Oracle services have emerged to connect blockchains to the real world. Originally popularized on Ethereum, the development of oracles led to a new wave of Web 3.0 innovations previously challenged by the lack of real-time external data feeds. Oracles have since become a key component of on-chain infrastructure, facilitating countless data queries each day across numerous blockchains. Today, oracles command a market with a total value locked (TVL) of over 88 billion USD [12].

**Motivation.** Oracle functionality relies on a combination of on-chain and off-chain components and actions that include listening to data requests, interacting with off-chain data sources, processing and aggregating responses, and the eventual response reporting. An assortment of different dApp needs led to a large number of oracles [13]–[25], [25]–[40], which vary based on their security and privacy guarantees, implementations, trust assumptions, and capabilities in terms of supported queries. Moreover, oracle designs range from theorized ones limited to academic works to those deployed and widely used in practice. Furthermore, for those deployed, the capabilities, overhead, and support differ between oracles' available documentations and what is actually deployed.

Indeed, oracle security and performance levels impact the dApps that use them. Reporting fraudulent or inaccurate external data feeds may have devastating impacts for dApps and their end users. Trading and lending protocols, for example, would execute trades or liquidate loans based off inaccurate prices. Supply chain management may retain a fake product to be original due to fraudulent oracle attestations. At the performance level, high response delays impact user experience, and may lead to dApp executions based on outdated data, e.g., trading based on old high prices. Thus, it is critical to understand how oracles operate and their security guarantees.

All of these factors complicate the landscape of oracles, which in turn makes it hard for researchers and practitioners to examine the space, accurately assess existing oracles and understand their trade-offs, or even build on/employ these services. As a result, this work is motivated by the following research questions:

**RQ1.** What capabilities (queries, data sources, etc.) do existing oracles support? and how do their designs and implementations impact these capabilities?
**RQ2.** What security and privacy guarantees do existing oracles support?
**RQ3.** For deployed oracles, what capabilities are currently being offered for dApps versus still being developed? and how performant are they in practice?
**RQ4.** What are the gaps and open problems that should be targeted so oracles can achieve their full potential?

**Contributions.** To answer these questions, we develop a systematization of knowledge for blockchain oracles. To the best of our knowledge, our work is the first to provide an extensive study of (academic and industrial) oracles while empirically investigating the capabilities of these deployed in practice. The utility of our work is three fold: First, simplifying the task for newcomers to understand the landscape. Second, offering a guideline for practitioners when choosing oracles for practical applications. Third, paving the way for new developments to address current gaps and expand the state of the art. In particular, we make the following contributions.

*Systematization framework.* We develop a multi-dimensional systematization framework for oracles covering three categories: *capabilities*, *security and privacy*, and *infrastructure*. To offer more granular insights, we introduce several sub-dimensions across these categories. These include query types and interaction patterns (for capabilities); trust/security assumption, authenticity, privacy, and economic aspects (for security and privacy); and architecture types and blockchain agnosticism (for infrastructure). We believe that our framework offers a holistic approach for the evaluation of existing and future oracles, enabling a clear path to understanding the landscape and distilling design, security, and capability trade-offs.

*Analyzing existing oracle services.* Leveraging our systematization framework, we analyze and categorize 24 academic and industrial oracles with a focus on popular and novel oracle designs. Our analysis identifies several key points related to various dimensions. For example, in terms of capabilities, there is a limited support for arbitrary queries and off-chain computations. In terms of response authenticity, we find that the majority rely on statistical aggregation or optimistic approaches, and only a few support proofs of authenticity. We also identify a trend of limited privacy support primarily pointed towards input privacy. This is in addition to variations in how an oracle is deployed in practice (for industrial oracles), and whether it can offer multi-blockchain support.

*Empirical evaluation.* To contextualize oracles' impact in practice, we conduct an extensive empirical study evaluating active industry solutions over popular smart contract-enabled blockchains. The goal is three fold: identify actual oracle usage and in which applications, understand the maturity of the ecosystem in terms of what capabilities are currently available, and quantify query overhead/cost and response delay. Our findings pinpoint the current status of deployed solutions, and provide insights on developers' experience when testing and using oracles, e.g., whether prior permission and help from the development team is needed to test particular capabilities.

*Distilling insights and identifying gaps.* Throughout our analysis, we distill numerous insights and call attention to a number of gaps. We find most oracles focus on particular query types, mainly price feeds, driven by current dApps' interests rather than building generic solutions. Moreover, there is very limited support for blockchains other than Ethereum. Furthermore, we discuss how relying on third parties or inter-blockchain messaging protocols require additional trust assumptions and may limit capabilities especially for multi-blockchain support. We also discuss oracle implications in terms of connecting blockchains to legacy systems, relation to MEV, potential misuses, and their interplay with AI. We believe that these insights, gaps, and discussions lay down directions for future work and offer a guideline for the use of existing oracles in practice.

**Related Work.** There are several SoKs and surveys around oracles. However, they either study a small number of oracles, or focus on these tied to a single blockchain or particular use cases, not to mention that many of them are out-of-date given the rapidly changing landscape.

In detail, Beniiche [41] studies oracle designs in terms of their interaction patterns, (de)centralization aspects, and data source types, while investigating only two oracles—Provable and Chainlink. Al-breiki et al. [42] focus on oracle design and components of trust, and Pasdar et al. [43] focus on establishing confidence in oracle responses, under two categories: voting-based and reputation-based approaches. Zhao et al. [44] study the design and trustworthiness of oracles used in DeFi, whereas Ezzat et al. [45] view oracles as an interoperability solution, comparing them to other blockchain interoperability approaches. Both these works examine a small set of oracles, in the range of $5-8$ solutions. Eskandari et al. [46] cover 17 oracles, studying the workflow steps and mechanisms underlying oracle operation, highlighting security risks for each. However, the selected oracles are all for Ethereum (excluding Band [29]), many of them are not general-purpose oracles, i.e., specialized like MakerDao [24], and even viewing protocols, like Uniswap [47], as oracles.

None of these works provide empirical evaluations of oracles in practice. There have been some empirical works (rather than surveys or SoKs) that studied particular oracle behavior in practice. Liu et al. [48] study price discrepancies between various oracles in DeFi, Gansauer et al. [49] examine the variations in price between Chainlink data feeds and centralized exchanges, Gangwal et al. [50] evaluate the internal accuracy of price feeds in Chainlink, and Cong et al. [51] examine whether or not having oracle integration impacts the underlying economic performance of DeFi dApps. While impactful, these studies are limited in scope highlighting only price reporting rather than offering a holistic study of various oracle solutions.

Our work stands out as it: (1) presents a granular systematization framework of oracles covering dimensions that have not been studied before such as deployment architecture, multi-blockchain support, and privacy, (2) analyzes an extensive list of 24 academic and industrial oracles spanning various blockchain architectures, and (3) conducts empirical evaluations investigating oracle capabilities and query overhead in practice, thus offering additional insights and identifying limitations related to practical deployments and their usability.

## 2. Background

To facilitate the discussion, we review the general design structure of oracles and their interaction patterns.

### 2.1. Oracle Components and Workflow

Oracles connect on-chain applications, or dApps, with the real world. As shown in Figure 1, an oracle consists of two main components: a *frontend* which is the on-chain component that faces dApps and allows them to interact with the oracle, and a *backend* that encapsulates most of the (off-chain) oracle workings needed for responding to on-chain queries.

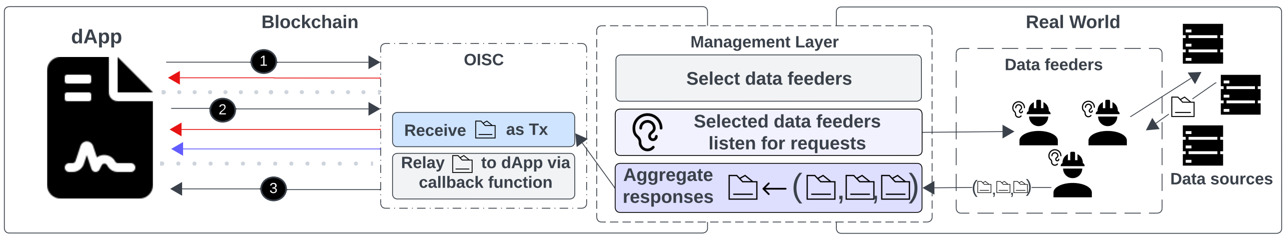

Figure 1. Oracle design architecture. dApps can interact with an oracle using the interaction patterns indicated by ball numbers: ❶ = request-response, ❷ = publish-subscribe, and ❸ = immediate-read.

**Frontend: Oracle interface smart contract (OISC).** Smart contracts can only interact with modules within the blockchain. Thus, an oracle must have an on-chain component, an oracle interface smart contract (OISC), deployed on the target blockchain. OISC specifies available query types, supported data sources, and oracle fees. It implements the interaction patterns that determine how dApps can submit requests and receive responses.

**Backend: Data feeders.** This is the far end of the oracle system that sits in the real world and interacts with data sources. Data feeders are parties that prepare responses for the oracle queries made by dApps. They can simply retrieve data feeds or further perform computations on-demand. These feeders are often rewarded for correctly fulfilled requests. Depending on the oracle design, feeders could be trusted entities operated by the oracle itself, or third parties who can join the (decentralized) oracle systems. Furthermore, in the latter, some oracles have a decentralized oracle network (DON), which is a network of the data feeders underlying the oracle.

**Backend: Management layer.** This comprises the middle layer sitting between the OISC and the data feeders. An oracle may contain additional modules for managing data feeders, such as registration and selection (in case multiple feeders are qualified to answer a particular request). This layer may also implement countermeasures to ensure integrity, such as verification of authenticity proofs. In addition, when multiple data feeders are tasked with fulfilling a request, this layer implements response aggregation (e.g., median, mean, quartile range of the retrieved data, etc.) to combine the results into a singular value. Furthermore, for oracles who have their own app-chains, thus they have validators (or miners) maintaining the oracle blockchain, those validators and all mechanics needed to manage them are part of the management layer.

**Workflow.** As depicted in Figure 1, a dApp initiates an oracle request by invoking the appropriate function in the OISC. This request contains input parameters (e.g., timestamp, URL source, etc.). Oracles may implement different interaction patterns (as we explain next), which impact the role of data feeders and how they are selected. Some oracles provide periodically-updated data feeds that dApps can read directly. In other instances, data feeders are selected on a request-by-request basis, so the management layer is the one who listens to new queries and then selects feeders to handle them. Moreover, some oracles preselect a set of feeders to listen and respond to all queries of particular type.

In either case, upon receiving a request, a data feeder starts preparing a response by querying the appropriate data source. Upon retrieving the requested data, the data feeder relays the response by submitting a transaction to the OISC. This triggers the OISC to send the response back to the dApp in a callback function. In case of multiple data feeders, upon receiving some threshold of responses, post-processing and aggregation will be done resulting in a singular response being sent to the OISC. Post-processing is commonly done off-chain in peer-to-peer networks of data feeders nodes, upon a consensus being reached on a singular value, one of the data feeders posts the final response to the OISC.

## 2.2. Oracle Interaction Patterns

Oracles implement various interaction patterns with dApps, thus allowing for various operational modes. Originally studied by Beniiche [41], there are three oracle interaction patterns (indicated by ❶-❸ in Figure 1):

❶ **Request-response.** This is a natural (reactive) interaction pattern; dApps send queries, by calling a function in the OISC, specifying the needed parameters, such as URL, dates, etc. Upon receiving a request, data feeders prepare the required response (after contacting the appropriate data sources) and send it back to the OISC, which in turn is sent to the dApp via a callback.

❷ **Publish-subscribe.** This is also a reactive pattern since the dApp submits a request to initiate the interaction. However, this request will result in multiple responses sent to the dApp instead of one. That is, the request determines the frequency or time periods during which the dApp will receive periodic responses from the oracle (indicated by multiple response arrows in Figure 1). Thus, a request acts a subscription for a series of future responses. Some models have the future responses triggered by updates instead of periodic time. For example, in price feeds, a new price is reported when a specified variance threshold between the last reported value and the current off-chain price is surpassed. In this case, subscriptions are typically set according to a budget (a deposit made by the dApp) representing the total fee amount a dApp is willing to pay, which determines the number of responses that will be received. This interaction pattern is commonly used for volatile data feeds, such as price and weather data, that change often and need to be updated regularly.

❸ **Immediate-read.** This pertains to data that is immediately available on-chain—stored in the OISC and automatically updated, usually in a periodic fashion, by a data provider. Although it does not require issuing requests, which generally reduces delays in obtaining real-world feeds, the resulting data may not be fresh, depend-

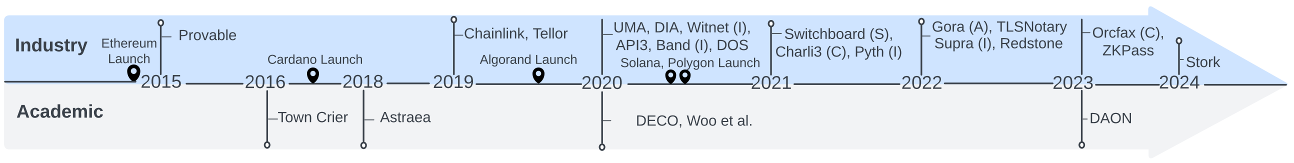

Figure 2. Evolution timeline of academic and industrial oracles.

ing on the period when it was last updated. This pattern can be beneficial for external data that by nature is being updated regularly, such as popular asset prices.

## 3. Systematization Methodology

**Scope and methodology.** Oracles make up a large body of work composed of peer-reviewed papers, whitepapers, and industry reports. We focus on prominent oracle designs including peer-reviewed academic works and popular industrial solutions. This paper does not attempt to describe in detail how these oracles work, but rather contextualizes their capabilities, design paradigms, key features, and (for industrial ones) examine how they work in practice.

For *academic works*, we used the search term “blockchain oracle” and examined the first 20 search result pages from Google Scholar, IEEE Explore, ACM Digital Library, and USENIX. Works that: (a) develop use case-specific oracles, (b) do not connect to the real world, (c) do not provide a full system design, e.g., improve only a specific component of an oracle system, (d) are closely related to existing designs, are out of scope.

Based on this scope, our refined search identified 5 academic oracles, including Town Crier [22], one of the first works on authenticated oracles via trusted hardware; Woo et al. [28] which extend Town Crier with a distributed version known as DiOr-SGX; DAON [52], a subsequent work that adds a reputation mechanism; Astraea [21] that proposes a stake-based approach to decentralization;[1] and DECO [23], a decentralized and privacy-preserving oracle.

For *industrial works*, our scope includes oracles operating on various blockchains, both EVM and non-EVM. We focus on active oracle solutions that have public documentation. In our search, we used a multi-faceted approach based off (a) oracle popularity in terms of their current and historical TVL, (b) varying architecture support, and (c) those with unique design features.

Current popular oracles were selected from the top 15 in terms of their TVL [12] filtered by having live deployments, leaving us with 10 oracles; Chainlink [17], Pyth [30], DIA [27], Redstone [37], Supra [39], Switchboard [38], API3 [35], Band [29], UMA [19] and Stork [40].[2] For historically popular oracles, i.e., those were popular in the past but their current TVL is not large, we included additional 4 oracles; Provable [16], Tellor [18], DOS Network [34] and Witnet [36].[3] Interestingly, among these 14 oracles, none is being operated on Cardano or Algorand, thus we include relevant oracles for both, finding 2 oracles for Cardano—Orcfax [56] and Charli3 [57], and 1 for Algorand—Gora [58]. To round out our examination, we finally select industrial works that feature unique oracle designs, selecting 2 works that offer unique response-authenticity techniques (i.e., authenticity proofs). These are TlSNotary [31] and ZkPass [26].[4]

Figure 2 traces the development timeline of these oracles, and for industrial ones, it includes their launch on popular blockchains. We specify launch on Ethereum without parentheses after an oracle name, while we use (A), (C), (S), (I) for Algorand, Cardano, Solana, and app-chains, respectively.

**Systematization framework.** We develop a systematization framework that distills design characteristics and distinguishing features of the studied oracles. Our framework covers the following three categories.

(1) Oracle capabilities. This category defines what *query types* an oracle can support, and how dApps can make requests and receive responses under what we call *interaction patterns and request location*. For query types, we divide them into three sub-dimensions: queries to predefined data sources, queries to arbitrary URLs, and off-chain computations. For interaction patterns, these include the patterns described in Section 2, and to offer a granular examination, we add the request location. That is, whether the query is recorded on the same blockchain where the dApp lives, on the oracle’s app-chain, or off-chain.

(2) Security and privacy. This category identifies the security assumptions/guarantees an oracle relies on/provides. It includes several sub-categories as follows.

*Security and trust assumptions*. Data feeder selection determines whether an oracle is centralized or decentralized, while the mechanisms used to ensure feeders’ honest behavior determine security/trust assumptions which include cryptographic assumptions, trusted hardware, trusted third parties, and economic security.

*Authenticity of responses* is about ensuring that an oracle provides correct and accurate query responses. We find that existing mechanisms include: reliance on a trusted party, optimistic approaches that employ response disputes, statistical aggregation, and authenticity proofs.

*Privacy*, which covers varying notions including user anonymity, input/output privacy, and function privacy. We discuss these notions, then focus on input/output privacy—the only notion considered by the surveyed oracles.

(3) Infrastructure. This category includes three dimensions: *oracle implementation architecture*, *blockchain agnosticism* or multi-blockchain support, and *oracle fees and rewards*. The implementation architecture can be split into: app-chain based, pure smart contract based, and a combination of smart contracts and off-chain modules. Blockchain agnosticism, on the other hand, is both an

1. The works [53]–[55] improved on the game-theoretics of Astraea’s model. Due to their similarity in design, we forgo discussing them.

2. As of July 2025, Chainlink, Pyth, and Redstone account for 74% of total oracle TVL, around 65.4 billion USD [12], while Band, ranked last on our list, has a TVL of around 150 million USD.

3. In the final stages of this work, we found that Provable and DOS were no longer active (no official announcements of deprecation).

4. We forgo Chronicle [25] since it only offers price feeds to a permissioned set of dApps, and Edge [59] due to limited documentation.

TABLE 1. ORACLE CLASSIFICATION BASED ON THEIR CAPABILITIES (◐ = CONDITIONAL SUPPORT).

| Oracle Capabilities | | Town Crier | DECO | DiOr-SGX | DAON | Astraea | Provable | Chainlink | Pyth | Band | API3 | UMA | Tellor | Witnet | DOS | DIA | Orcfax | Charli3 | Gora | ZKPass | TLSNotary | Switchboard | Stork | Supra | Redstone |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Query Types** | Pre-defined data feeds | | | | | | | ● | ● | ● | ● | | | ● | ● | ● | ● | ● | ● | ● | | ● | ● | ● | ● |
| | Arbitrary queries | ● | ● | ● | ● | ● | ● | ● | | ● | | ● | ● | ● | | | | | ● | ● | ● | ● | | | |
| | Off-chain computations | | | | | ◐ | ● | ● | | ● | | ◐ | ◐ | | | | | | ● | | | | | | |
| **Interaction Patterns** | Publish-subsribe | | | | | | | ● | | | ● | | ● | ● | | ● | | | | | | | | | |
| | Request-response | | | | | | ● | ● | ● | ● | ● | ● | ● | ● | ● | | | | ● | ● | ● | ● | ● | ● | ● |
| | Immediete-read | | | | | | | ● | | | | | | ● | | | ● | ● | ● | | | | | ● | ● |
| **Request Location** | DApp blockchain | | | | | | ● | ● | | | ● | ● | ● | | ● | ● | ● | ● | ● | | | | | | |
| | Oracle blockchain | | | | | | | | | ● | | | | ● | | | | | | | | | | | |
| | Off-chain | | | | | | | | ● | | | | | | | | | | | ● | ● | ● | ● | ● | ● |

implementation detail and an attractive feature. That is, an oracle is implemented in a way that can seamlessly work with several blockchains. We find that mechanisms used to support this property include employing inter-blockchain messaging protocols and deploying separate oracle instances. Lastly, we discuss the various fee and rewards structures used by oracles, i.e., how they reward data feeders and what/how dApps pay for oracle queries.

## 4. Oracle Capabilities

Being primarily a tool for bringing real-world data on-chain, we begin with categorizing the examined oracles based on their capabilities primarily defined as the query types they support. Furthermore, there are other design factors that impact these capabilities, including interaction patterns and request location. We evaluate the examined oracles based on three dimensions (Table 1).

### 4.1. Query Types

This dimension encompasses not only what types of real-world data/computation can be accessed, but also query structure in terms of its format (arbitrary questions/-computations formulated by the dApp or a choice among a pre-specified data feeds), as well as data sources (arbitrary chosen by the dApp or a pre-defined set chosen by the oracle). Thus, we classify query types into *pre-defined data feeds*, *arbitrary queries*, and *off-chain computations*.

**Pre-defined data feeds.** This is the simplest and most popular query type category. It enables a dApp to request data from a pre-defined set of sources approved by the oracle. Each of these sources may support one or more data feeds, e.g., prices of numerous assets. Each oracle has its own procedure for managing the data sources, often selecting popular and trusted ones, along with adding new sources in line with strategic partnerships with emerging dApps. Pre-defined data feeds have been popularized for public and routine data, such as financial and weather data, based on a simple query format; a dApp specifies the source and the data feed of interest.

Oracles, since their inception, have been mostly used to provide real-world price feeds, and they still use pre-defined data feeds to do so. We find that 15 of the 19 industry solutions follow this approach (as shown in Table 1). Some oracles also report common economic metrics and indexes in addition to prices. In particular, Chainlink provides customer price index (CPI), real GDP, personal consumption expenditures (PCE) price index, real final sales to private domestic purchases. Also, Pyth provides CPI, PCE, the producer price index (PPI), employment and hourly wage growth rates, and lending rates such as U.S government bond yield and bank funding rate.

**Insight 1.** *Pre-defined data feeds unify the source and received feeds among dApps. This is highly appreciated in applications such as trading to avoid huge discrepancies that may harm end users.*

The specific data available varies from oracle to oracle. As expected, widely used data types are usually available from all oracle price feeds, e.g., BTC/USD price. Also, for oracles with multi-blockchain support (see Table 4), their pre-defined data sets may vary depending on the blockchain being targeted. Chainlink, Redstone, and Witnet all have varying number of data feeds per blockchain. Respectively, Chainlink has 183, Redstone, 70, and Witnet 1 immediate-read data feeds available on Ethereum, whereas 142, 4, and 14 on Polygon, and $212, 22$, and 2 on Arbitrum. This fluctuations continues to their full set of supported blockchains. These decisions are highly impacted by the mechanisms used for multi-blockchain support, which we discuss in Section 6.2.

**Insight 2.** *Due to design dependencies, pre-defined data feeds availability could be fragmented not only among different oracles, but also within the same oracle among different blockchains.*

**Arbitrary queries.** This class enables a dApp to request data from an arbitrary source; either an arbitrary URL (web-based) or any party that can answer a question of interest (human-based). It is a flexible class usually used for unpopular/new feeds, e.g., prices of emerging digital assets or results of sport events.

**Insight 3.** *Arbitrary queries can handle trust issues; a dApp, that does not trust the pre-defined data sources, can specify its desired source instead.*

*Web-based* arbitrary queries are the most common class taking the form of HTTP requests to arbitrary URLs. A query contains the source URL, along with parameters specifying the data type/content a dApp wants. This class is offered by 15 oracles as shown in Table 1.

*Human-based* arbitrary queries utilize human-formulated requests and responses. They are usually used for event-based data, as in prediction markets [60], where data is publicly known but not easily accessible via pre-defined sources. A request is formatted as a clear and publicly-verifiable question that anyone can answer. As this paradigm generally adopts the optimistic approach with a dispute resolution to challenge incorrect responses, those who want to respond must put in stake that will be slashed if their responses found to be incorrect. Only 3 oracles support the human-based class—Astraea, Tellor, and UMA. Tellor by design supports arbitrary queries, however, in practice, responders are limited to a pre-authorized set. While highly flexible, human-based queries introduce new challenges related to whether a request formulation is clear, and subsequent disputes over the wording of a question and the respective response.

**Off-chain computations.** Under this class, oracles facilitate expensive computations off-chain, to avoid the expensive on-chain cost, while relaying the output to the requesting dApp. A computation query is composed of a script and its inputs. To reduce on-chain storage, the script is either posted on external storage sites, e.g., IPFS [61], or sent directly to the data feeders. The inputs also can be passed-in parameters in the query or retrieved via an external call to a data source.

We find that 7 oracles support off-chain computations (Table 1).[5] In principle, human-based oracles—UMA, Tellor, and Astraea—can support off-chain computations; an arbitrary question can contain a computation steps along with the desired inputs. However, only computations that can be publicly done in a deterministic manner are supported. These oracles may further limit the output type; Astraea supports T/F questions, thus it is limited to computations with binary output. Whereas UMA and Tellor support both T/F and numerical responses, thus offering a wider range of computations. Computation questions, however, must be carefully formulated as the human-interpretable nature may lead to new ambiguities not found in scripts. Table 1 denotes this conditional capability with a half-shaded circle.

Provable, Chainlink, Band, and Gora support off-chain computation in a more explicit way. They require particular scripting languages: C++ for Provable, Javascript for Chainlink, Rust for Band, and WebAssembly for GORA. Both Chainlink and Gora rely on sending the script directly to a P2P network of data feeders (called a decentralized oracle network—DON) responsible for such requests. While Provable and Band require scripts to be uploaded to IPFS and Band's app-chain, respectively, to be referenced in computation queries. In general, sending scripts directly to data feeders may limit code re-use and context needed to understand computation results—data feeders are not required to hold the script permanently.

**Insight 4.** *We believe that the increased adoption of layer 2 scalability solutions, e.g., rollups [63]–[66], and recent developments of co-processors that enable performing expensive computations off-chain [67], have shifted the focus of oracles away from supporting computation queries.*

## 4.2. Interaction Patterns

A related aspect to oracle capability is the interaction pattern facilitating queries. We note that academic oracles focus on the mechanisms for retrieving off-chain data without specifying explicit interaction pattern, although it can be implied that they follow the request-response one. Thus, we focus on industrial oracles.

5. Truebit [15] is a prior oracle for off-chain computations. However, this older version is inactive (a new rendition is being developed [62]).

Although each query type can be facilitated using any of the interaction patterns discussed in Section 2, we find that immediate-read patterns are primarily used for pre-defined data feeds. When an oracle offers multiple interaction patterns, pattern selection is influenced by the underlying dApp use cases, e.g., response frequency or that certain data types may be only available via certain patterns. In general, for oracles, interval-based patterns (publish-subscribe and immediate-read) cost more resources than request-response; data is routinely updated on-chain regardless of whether it is actively being utilized.

The majority of the examined oracles support the request-response style (all except Charli3 and Orcfax). Chainlink, Witnet, Orcfax, Gora, Redstone, Supra, and Charli3 offer price feeds using the immediate-read pattern. Moreover, Chainlink, Witnet, Tellor, API3, and DIA, provide publish-subscribe style responses, primarily for financial data feeds (API3 further offers it for weather/geographic data). Furthermore, and naturally, off-chain computations employ the request-response pattern. Lastly, interaction patterns may vary based on the underlying blockchain, in the case of Chainlink, certain patterns like immediate-read are not available on all the supported chains, for instance, only the pull-based request-response pattern is available on the Solana, Sei [68], and Unichain [69] blockchain.

**Insight 5.** *Interaction patterns are tied directly to the specific data type being offered. Namely, immediate-read patterns are only offered for pre-defined data feeds. Additionally, oracles with multi-blockchain support may adopt different patterns across different blockchains.*

## 4.3. Request Location

In patterns that require explicit requests, another distinguishing factor is where these requests are recorded. These include: (1) *on-chain: dApp blockchain*, where requests are published on the blockchain where dApps live, (2) *on-chain: oracle app-chain*, where requests are relayed to the oracle app-chain and recorded there, and (3) *off-chain* where queries happen off-chain.

**On-chain: dApp blockchain.** Since the OISC is deployed on the blockchain where dApps live, all calls (encapsulating requests) made to OISC are published there. As shown in Table 1, we find that 10 oracles follow this approach. Under multi-blockchain support, this location instance means that a separate OISC is deployed on each blockchain and requests are published accordingly. This model is denoted by **1** in Figure 3.

**On-chain: oracle app-chain.** Many oracle services started to deploy their own blockchains, i.e., app-chains. To communicate with dApps deployed on general-purpose blockchains, inter-blockchain messaging protocols are used. These could be specialized protocols operated by the oracle itself as in Witnet, or general purpose solutions such as IBC [70] as used by Band. Only Band and Witnet fall under this category of request location.

Accordingly, a dApp sends the request to the app-chain using the messaging protocol, with the responses prepared by the data feeders are recorded on the app-chain. Relaying the response back to the dApp works

TABLE 2. ORACLE CLASSIFICATION ACROSS SECURITY AND PRIVACY DIMENSIONS.

| Security and Privacy | | Town Crier | DECO | DiOr-SGX | DAON | Astraea | Provable | Chainlink | Pyth | Band | API3 | UMA | Tellor | Witnet | DOS | DIA | Orcfax | Charli3 | Gora | ZKPass | TLSNotary | Switchboard | Stork | Supra | Redstone |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Trust Assumptions | Cryptographic security | ● | ● | ● | ● | | ● | ● | | | | | | | | | | | | ● | ● | ● | ● | ● | ● |
| | Economic security | | | | ● | ● | | ● | ● | ● | | ● | ● | ● | ● | ● | | | ● | | | ● | | ● | ● |
| | Hardware security | ● | | ● | ● | | ● | | | | | | | | | | | | | | | ● | | | |
| | Trusted party | ● | ● | | | | ● | | | | ● | | | | | | ● | ● | | ● | ● | | | | |
| | Consensus | | | ● | ● | | | ● | ● | ● | | | | ● | ● | ● | | | ● | | | | ● | ● | ● |
| Centralization | Decentralized | | ● | ● | ● | ● | | ● | ● | ● | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● |
| | Centralized | ● | | | | | ● | | | | ● | | | | | | | | | | | | | | |
| Response Authenticity | No authenticity | | | | | | | | | | ● | | | | | | ● | ● | | | | | | | |
| | Dispute phase | | | | | ● | | ● | | | | ● | ● | | | | | | | | | | | | |
| | Aggregation method | | | | ● | | | ● | ● | ● | ● | | | ● | ● | ● | ● | ● | ● | | | ● | ● | ● | ● |
| | Authenticity proof | ● | ● | ● | ● | | ● | | | | | | | | | | | | | ● | ● | ● | | | |
| Privacy | No privacy | | | ● | ● | ● | | | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | ● | | | | ● | ● | ● |
| | Input privacy | ● | ● | | | | ● | ● | | | | | | | | | | | | ● | ● | ● | | | |
| | Output privacy | | ● | | | | | | | | | | | | | | | | | ● | | | | | |

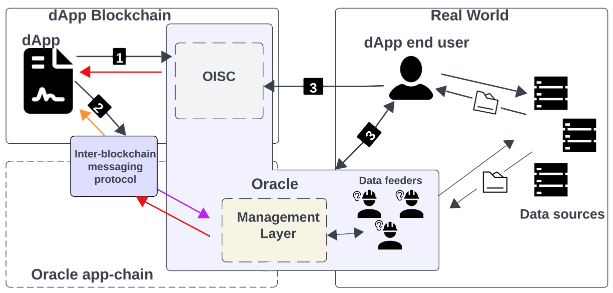

Figure 3. Oracle model by request location; dApp blockchain **1**, oracle app-chain **2**, and off-chain **3**.

pretty similar to a normal callback but facilitated by the messaging protocol (so the dApp should invoke the proper calls for the messaging protocol). This model is represented by **2** in Figure 3.

**Insight 6.** *App-chains promote multi-blockchain support but could increase overhead; messaging protocols require additional transactions (to relay requests/responses) leading to excessive delays.*

**Off-chain requests**. In some instances, end-users are aware that some data is needed for an on-chain dApp interaction, and are able to preemptively request this data off-chain and provide on-chain responses. For example, for a dApp that requires age-verification, the end-user can retrieve this data off-chain, then provide it along with some authenticity guarantee to the dApp. Off-chain request model (denoted by **3** in Figure 3) is commonly used when data correctness can be verified on-chain.

We find that 8 oracles follow this paradigm (Table 1). TLSNotary, ZkPass and DECO develop a third-party verifiable TLS variant to authenticate data feeds. While Pyth, Redstone, Stork, Supra, and Switchboard utilize a pull-based approach; feeds are automatically posted/updated on an external data layer, so requests can directly pull responses from this layer (Charli3 and Chainlink have recently announced a future upgrade offering pull-based oracles [71], [72]). For Pyth, data feeds are posted on its app-chain, Pythnet, and sent to Wormhole [73], an inter-blockchain messaging protocol, that listens and attests to seeing these values on Pythnet. End users can retrieve these feeds and their attestations, and send them to their dApps. Pyth further streamlines this retrieval process using an off-chain web-service, Hermes [74].

## 5. Oracle Security and Privacy

Oracles are data bridges between dApps and the real world. Thus, their security is tied to authenticity and integrity of the reported responses. That is, ensuring that the original request is not manipulated, that data feeders retrieve correct responses and/or perform computation correctly, and that they relay correct responses back. Indeed, examining oracle inner workings defines critical factors around their security and privacy. In particular, we want to answer questions related to the security/trust assumptions an oracle makes on the employed modules/parties, response authenticity, whether any form of privacy is offered, and how economic security (if employed) plays a role in all of these factors. We analyze the surveyed oracles based on these dimensions (Table 2); this is after discussing existing oracle security notions in the literature.

### 5.1. Oracle Security Notions in the Literature

Prior work lacks a generic formal definition of oracle security. DECO [23] introduced a notion for decentralized oracles that utilize third party verifiable TLS. The notion captures a prover $\mathcal{P}$, which is the data owner, i.e., a user of the dApp, who wants to prove some features about their private data, and a verifier $\mathcal{V}$, which is a third party verifying the authenticity of the data retrieved by $\mathcal{P}$ from a data source. The notion focuses on two properties: *prover integrity*, so a malicious $\mathcal{P}$ cannot forge content provenance, or convince $\mathcal{V}$ to accept incorrect response or reject valid responses. And *verifier integrity*, where a malicious $\mathcal{V}$ cannot cause $\mathcal{P}$ to get incorrect responses (all what $\mathcal{V}$ can do is not verifying the response authenticity before being used by the dApp).

Town Crier [22] presents an ideal functionality modeling oracle security, but it is specific to a trusted hardware-based design. This notion captures three security properties: *response authenticity*, *gas sustainability* ensuring users pay only for correctly-fulfilled queries (and the fees for their subsequent response transactions), and *trusted code base minimization*, that aims to minimize the trusted code base (executed in the TEE). In addition to being TEE-specific, the latter property is an implementation detail rather than a generic security property.

For oracles that have additional modules in their management layer, additional security properties are required. For example, an oracle app-chain must satisfy liveness and safety [75], [76]. Moreover, request/response privacy is another property of interest, thus, an oracle security notion needs to capture that as well.

**Gap 1.** *Defining a generic and formal security notion for oracles is an open question.*

## 5.2. Trust and Security Assumptions

An oracle involves various modules and parties; management layer, data feeders, and data sources. Sources are usually trusted, and to weaken this assumption, aggregation techniques can be utilized. The situation is different for the management layer and data feeders, and this section discusses the security and trust assumptions the examined oracles have around them.

**Centralized vs. decentralized.** Centralized oracles operate their own trusted data feeders, while decentralized oracles allow third parties to join as feeders. We find that 3 oracles are centralized: Provable, API3, and Town Crier, while the rest are decentralized. Decentralized oracles naturally impose some restrictions, such as minimal resources or financial (staking) requirements to be able to join the consensus layer. Also, specific affiliations might be required in order to join; Pyth requires data feeders to be financial institutions (firms, exchanges, etc.).

**Security countermeasures and assumptions.** Due to the power tied to the data feeders' role in reporting real world data to dApps, oracles focused on implementing countermeasures to thwart any misuses of such power.

*Cryptography*. Naturally, cryptography is widely used across oracle designs. Digital signatures are inherent to preserve integrity. Additionally, Chainlink, DIA, DOS, DAON, Supra, Redstone, Stork and Gora employ signatures for response authentication within their DONs (to indicate that data feeders agree on the reported response). Town Crier and DiOr-SGX utilize the default TLS protocol with its cyrptographic primitives, while DECO, TLSNotary, Provable and ZKPass construct a third-party verifiable variant that employs a two-party MPC protocol—also, Provable uses TLS 1.0-1.1 while the rest support TLS 1.2 and 1.3. Lastly, DECO and ZKPass employ zero knowledge proofs (ZKPs) to support response privacy.

*Trusted hardware.* Trusted execution environments (TEEs), or simply trusted hardware, are used to perform critical operations, e.g., enforce the correct behavior of data feeders. Only 4 oracles—Town Crier, DiOr-SGX, DAON, and Switchboard—employ this approach. Town Crier lets its data feeder execute TLS within a TEE, and upon completing the response which will be signed by the TEE, it is then relayed on-chain. DAON, Switchboard, and DiOr-SGX present a decentralized variant consisting of numerous data feeders each with its own TEE. The key idea is that while TEEs can enforce correct behavior, they cannot enforce feeder participation. By having multiple feeders respond to a query, it is more likely that at least one will participate. This could further be combined with aggregation techniques to produce a singular value when multiple responses are obtained. The use of TEEs offers strong authenticity guarantees, but this comes with strong assumptions on resistance of reverse engineering and side-channel attacks; several works [77]–[82] have demonstrated that TEEs are vulnerable to such attacks.

*Trusted party.* In some instances, data feeders are trusted. API3 falls under this category, which acts as a service of standalone first-party-oracle, i.e., trusted data sources run data feeders so these sources directly bring data on-chain. While not a conventional design choice, this approach reduces the trust to only the data sources—where most oracles assume so—without adding feeders in the middle who may misbehave.

TABLE 3. CLASSIFYING ORACLES (THAT EMPLOY STAKING) BASED ON ECONOMIC SECURITY ASPECTS (NS = NOT SPECIFIED). UMA HAS AN ORACLE-OPERATED ERC-20 TOKEN, CALLED UMA, USED FOR GOVERNANCE, WHICH CAN BE USED FOR THE COLLATERAL.

| **Oracle** | **Economic Security** | |
|---|---|---|
| | Asset of Collateral | Penalty Type |
| Astraea | N/S | Variable |
| DAON | N/S | Other |
| Chainlink | LINK | Fixed |
| Pyth | PYTH | Percentage |
| Band | BAND | Percentage |
| Tellor | TRB | Fixed |
| Witnet | WIT | Variable |
| DOS | DOS | Fixed |
| DIA | DIA | NS |
| Gora | GORA | Other |
| Switchboard | SWTCH/JitoSoL | Other |
| Supra | SUPRA | NS |
| Redstone | ETH/RED | Other |
| UMA | Any ERC-20 | Fixed |

**Insight 7.** *First-party oracles simplify oracle design, but limit data sources to those willing to deploy the on-chain data reporting functionality. This tightly managed design can be attractive to data providers with restrictive access control policies or legacy systems that want to begin offering data on-chain.*

*Economic security.* Many oracles rely on economic security to incentivize honest behavior of data feeders. A feeder puts in a collateral, when joining the system, that will be slashed for incorrect responses or for lack of activity (with an eventual removal from the system if a feeder continues to misbehave). As shown in Table 2, 14 oracles utilize this approach with various staking methods. For Astraea, Chainlink, DAON, Switchboard, UMA, Tellor, DOS, DIA, and Gora, staking is implemented within the OISC. While Redstone utilizes Eigenlayer's [83] restaking service on top of Ethereum, enabling Ether or a liquid staking token (RED) to be used. For app-chain oracles, namely Pyth, Band, Supra, and Witnet, this is integrated within the staking utilized in their blockchains. The digital asset of the collateral varies between oracles, as shown in Table 3, and generally it is an oracle-operated token.

**Insight 8.** *Economic security is challenging; it requires modules for defining/detecting misbehaviors, proper penalty setup, and in multi-chain oracles, the security of assets residing on different chains.*

Defining what constitutes an accurate response could be difficult. For example, reporting world events (e.g., election results) is publicly verifiable, while other responses (e.g., answering human-based requests) may not have an agreed-upon value due to ambiguities in request formulation. Moreover, reporting volatile data (e.g., digital asset prices) may lead to accuracy issues—its value might change between the time of retrieval and time of delivery.

Another important factor is the financial penalty bound required to deter malicious actors. We find that this also varies across oracles (see Table 3); a fixed penalty amount, as in Chainlink, Tellor, UMA, and DOS, a percentage of stake, as in Switchboard, Pyth and Band (the latter two slash 5% of stake), or forgoing direct penalties for stopping future revenue temporarily (and eventual removal) as in Redstone, DAON, and Gora. Moreover, penalties may vary within the same oracle based on the data feeds. Chainlink currently only penalizes misbehavior for a singular price feed, the immediate-read ETH/USD feed, with plans to incorporate penalties into additional feeds. For UMA and Witnet penalties are determined on a feed-by-feed basis; a data feed creator determines what if any penalties to utilize.

**Gap 2.** *Oracle designs that employ economic security lack formal game theoretic analysis proving that their stake/penalty setting enforce honesty. In fact, only three oracles—UMA, Astraea, and Chainlink—offer informal arguments for their configurations.*

### 5.3. Response Authenticity

Authenticity pertains to the accuracy of the returned response compared to the truth value in the real world. We find that authenticity mechanisms employed by the examined oracles fall under four categories: *no authentication*, an *optimistic approach* with dispute resolution, *statistical aggregation*, and explicit *authenticity proofs*.

**No authentication.** Authentication is not needed when data feeders are trusted. Only API3 follows this approach due to its first-party oracle design discussed earlier.

**Optimistic approach.** We find that 3 oracles, Astraea, UMA and Tellor, employ this approach, which is used entirely for publicly verifiable data. This paradigm presents an efficiency-finality trade-off; no explicit authenticity proofs are needed, but a response is final only when the dispute period is over. In detail, a data feeder's stake will be slashed if a dispute (a claim from another party that a response provided by this feeder is incorrect) is successful during the dispute phase following each response reporting. An arbitration period starts when a dispute is submitted, where additional parties are brought in to reevaluate the response and agree upon an outcome. Tellor relies on arbitration from its staked data feeders, Astraea has additional parties called certifiers to raise disputes and handle arbitration, while UMA has its digital asset token holders weigh in the arbitration process.

Optimistic approaches rely on economic means to incentivize parties; financial penalties against misbehaving feeders, and rewards for parties that solve disputes. In general, this is similar to the underlying paradigm of optimistic rollups [84], [85]. As in rollups, incentive compatibility of the verifiers is a critical issue; if parties are not incentivized to vet submitted results and raise complaints, then incorrect reported results would be accepted.

**Insight 9.** *Configuring the optimistic approach could be challenging. Critical data may require strong incentives for verifiers, whereas highly volatile data may necessitate short dispute periods—thus risking accuracy.*

**Statistical aggregation.** Instead of relying on a single data source/feeder, this technique aggregates a singular response from a set of data feeders querying varying data sources. The hope is that aggregation can isolate invalid/inaccurate responses or reduce their impact on the final response. We find that 14 oracles (Table 2) utilize this approach (for Switchboard and API3, while aggregation from multiple-data sources is offered, the default is retrieving data from a single source unless otherwise specified.). Consensus on the aggregation is either reached off-chain in the oracle DON as in Chainlink, DOS, DIA, Charli3, DAON, Orcfax, Gora, Stork and Redstone, or on-chain as in Pyth, Band, Witnet, Supra, and API3.[6] For off-chain aggregation, one of the data feeders in the DON is selected to post the aggregated response on-chain.

Various aggregation methods, such as mean, median, inter-quartile ranges, etc., are used to compute a single response. The median is the most common method as it is resilient to movement by a single outlier. Chainlink, DIA, Witnet, Band, API3, Redstone, and Stork enable dApps to choose the aggregation method at the time of request. On the other hand, Pyth offers only inter-quarter ranges to report a range of values within a confidence interval, while Orcfax and Supra only support the median.

**Authenticity proofs.** These are usually used when a single data feeder is tasked with responding to a query. The feeder provides a proof that the data it relayed from the data source has not been altered. Only 8 oracles support authenticity proofs as shown in Table 2. Town Crier, DioR-SGX, DAON, and Switchboard rely on the authenticity guarantees of TLS in conjunction with TEEs, so the TEE's signature is the proof. When the response is shared on-chain, the OISC, that has the TEE's public key hardcoded, verifies the signature. Whereas TLSNotary, DECO, and ZKPass employ versions of verifiable TLS in which the TLS session keys are distributed among a prover (the user of the dApp) and a third party verifier (as they employ the off-chain request model where the end user of a dApp provides attested responses).

Provable allows dApps to choose between TLSNotary, Android proofs, and Ledger proofs. Android Proofs is a combination of Google's software attestation tool Safety net (now deprecated [87]) and Android hardware attestation tooling. Ledger proofs rely on a type of hardware wallets called Ledger [88] to attest to code execution by a TEE on this wallet.

**Insight 10.** *Authenticity proofs offer the strongest authenticity guarantees but at the expense of increased complexity, high cost due to using cryptography, or strong trust assumptions (such as TEEs).*

### 5.4. Privacy

Oracle privacy is a multi-faceted question with varying considerations often based on individual oracle design and architecture. Foremost, input/output (I/O) privacy is about hiding the parameters of a request and/or the response. For off-chain computation queries, there is function privacy that aims to hide the computation itself. Lastly, privacy may also cover anonymity, thus hiding the identity of the

6. Supra utilizes a technique from [86] that employs a heuristic for aggregation requiring consensus between only a simple majority of the data feeders instead of a super-majority as in previous works.

TABLE 4. ORACLE CLASSIFICATION ACROSS ARCHITECTURE TYPES AND MULTI-BLOCKCHAIN SUPPORT.

| Infrastructure | | Town Crier | DECO | DiOr-SGX | DAON | Astraea | Provable | Chainlink | Pyth | Band | API3 | UMA | Tellor | Witnet | DOS | DIA | Orcfax | Charli3 | Gora | ZKPass | TLSNotary | Switchboard | Stork | Supra | Redstone |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Architecture Types | App-chain | | | | | | | | ● | ● | | | | ● | | | | | | | | | | ● | |
| | SC only | | | | | ● | | | | | | ● | | | | | | | | | | | | | |
| | SC + off-chain components | ● | ● | ● | ● | | ● | ● | | | ● | | ● | | ● | ● | ● | ● | ● | ● | ● | ● | ● | | ● |
| Blockchain Agnosticism | Single-chain support | | | | | | | | | | | | | | | | ● | ● | ● | | | | | | |
| | Seperate implementations | | | | | | ● | ● | | | ● | ● | ● | | ● | ● | | | | ● | ● | ● | ● | | ● |
| | Inter-blockchain messaging | | | | | | | | ● | ● | | | | ● | | | | | | | | | | ● | |

end-user, or the dApp, that made the request and/or the data feeder who responded. The security notion in [23] defines a privacy property, stating that a malicious $\mathcal{V}$ only learns public information about $\mathcal{P}$'s requests and the evaluation result, thus it only covers I/O privacy.

The majority of examined oracles, 17, do not support any privacy (Table 2), and those who do are limited to I/O privacy. As for user anonymity, we argue that for on-chain requests, this is a consideration for the blockchain where dApps live; a request is made by issuing a transaction so anonymity is governed by the underlying blockchain. Indeed, for off-chain requests and data feeder anonymity, the oracle needs to ensure anonymity, which has been absent in the examined oracles. Accordingly, below we discuss I/O privacy support.

**Gap 3.** *Privacy is not a focus for most oracles and, when considered, only applies to input/output and a subset of query types.*

**Input privacy**. This is desired for arbitrary queries and off-chain computations. One reason is the prevalence of authenticated data sources, which forces a party requesting data to pass (personal) authentication material when calling the data source API (e.g., to check that the requester has paid subscription to request data). Thus, the user/dApp must provide this material to the oracle so it can request the data. Subsequently, a dApp wants to hide this sensitive material from the oracle, and in some instances, the entire request to avoid deducing information about the dApp operation/decisions upfront.

Only 4 oracles—Chainlink, Provable, Switchboard, and Town Crier—support input privacy. Chainlink employs threshold encryption; the request is encrypted under a DON's public key with the decryption key shared among the data feeders in this DON. Upon sending an encrypted request, selected feeders decrypt it then respond, thus there is an implicit assumption that those feeders will not disclose the request to the public. Provable allows users to encrypt part of there requests under the public key of its centralized data feeder. It, however, supports hiding only one input—usually used for authentication material. Finally, Town Crier and Switchboard requests can be encrypted under the TEE's public key, so only the TEE can decrypt and respond.

**Output privacy.** This is relevant when the response has information a requester wishes to hide. Consider an oracle used for identification verification purposes. Here, the oracle may only need to partially reveal information about the response, i.e., attesting that identification meets certain requirements, without revealing the full response.

DECO and ZKPass are the only oracles offering output privacy using ZKPs. ZKPass employs a VOLE-based interactive ZKP [89], [90], whereas DECO does not specify a certain scheme. Regardless of the adopted proof system, these oracles follow the same general verification paradigm. The end-user (of a dApp requesting data) generates a ZKP to be verified by a data feeder. This verification can then be relayed on-chain to a verification contract operated by the dApp in conjunction with the oracle; in a sense, what is relayed is a signature from the data feeder over the response and an attestation from the oracle that this feeder is authorized to do so. Such smart contracts are necessarily use case-specific encoding conditions and information regarding a certain statement.

## 6. Infrastructure

We discuss issues related to oracle infrastructure covering deployment architecture, blockchain-agnosticism or oracles' ability to service multiple blockchains, as well as fees and rewards models. Table 4 classifies the examined oracles based on architecture and blockchain-agnosticism, whereas Table 5 classifies them based fees and rewards type. Studying these dimensions is beneficial for future oracle design, or improving existing ones if possible, in terms of producing flexible and interoperable oracles and understanding their economic aspects.

### 6.1. Architecture Types

We find that oracle architecture takes on three forms: app-chains, pure smart contract deployment on a general-purpose blockchain, and a combination of smart contracts and off-chain components (indeed all oracles involve off-chain components—data feeders and sources, here we mean additional off-chain modules on top of these).

**App-chain based.** An app-chain is a blockchain centered around implementing the oracle functionality and its management layer. Interaction with this app-chain, to receive requests and relay responses, is done using inter-blockchain messaging protocols. We find 3 oracles (Witnet, Pyth, and Band) operate as PoS app-chains (Tellor recently announced Tellor Layer [91] a planned app-chain within Cosmos [92]). The app-chain validators could be also tied to the oracle modules (rather than just running the consensus/network protocol). Both Witnet and Pyth enact data feeders directly into the role of validators, while Band separates these two roles. In both cases, a consensus protocol is utilized for response aggregation.

**Smart contract based.** Here, the entire oracle functionality is implemented within the OISC deployed on a smart contract-enabled blockchain. This includes data feeder selection, response aggregation (if any), response relaying to dApps, and reward distribution and/or handling staking and financial penalties. This design is found to be common for oracles supporting human-based requests—UMA and Astraea. Smart contract-based oracles lead to more on-chain transactions as all interactions happen on-chain, which adds to the scalability issue of blockchains.

**On-chain/off-chain module combination.** This approach improves efficiency by handling costly operations off-chain. Oracles implement most of their functionality within the OISC as above, but also have external tooling/-modules for data retrieval. These include consensus in a DON, specialized software for data retrieval, or trusted hardware for secure data retrieval/computation.

We find that 20 oracles adopt this architecture (Table 4). Chainlink, DAON, Gora, DOS, DIA, Redstone, Supra, Stork, Orcfax, and Charli3 employ off-chain DONs to handle feeder selection and response aggregation. Operating a DON is less costly than operating an app-chain; a DON is considered a small-scale module compared to a full blockchain. For other instances—Provable, ZKPass, TLSNotary, DiOr-SGX, Town Crier, API3, Tellor, and Switchboard—the use of off-chain modules is limited to data retrieval and authentication. We note that oracles employing statistical aggregation (Chainlink, Gora, DOS, DIA, Orcfax, Charli3, Stork, and Redstone) work primarily in DONs due to the overhead of needing a large data feeder set to fulfill a request.

**Insight 11.** *Compared to other architecture types, app-chains minimize on-chain overhead and offer more flexibility to enable core oracle design changes and feature testing. In parallel, being independent blockchains, their design leads to additional complexities in terms of incentives and interoperability.*

## 6.2. Blockchain Agnosticism

Blockchain agnosticism, or multi-blockchain support, is about oracle's ability of supporting dApps deployed on any smart contract-enabled blockchain. While in theory we believe that all oracles can service any blockchain, in practice the majority naturally target Ethereum. Academic oracles do not specify their particular blockchain support as their designs are posed in a generic way. As shown in Table 4, of the 19 industry oracles, 17 offer some form of multi-blockchain support, while the rest, Charli3 and Orcfax, are only for Cardano. Furthermore, one may expect an oracle to offer the same capability level to all supported blockchains; we find that this is not the case.

**Gap 4.** *True blockchain-agnosticism of unilateral equal support is still an open question. Current attempts offer useful insights towards this goal.*

We find that the examined oracles support blockchain agnosticism using two paradigms: inter-blockchain messaging and multi-instance deployment.

**Inter-blockchain messaging protocols.** These are protocols that allow different blockchains to communicate. An oracle can use them to interact with dApps on any blockchain, of course by implementing the messaging protocol interfaces in, e.g., OISC. Even oracles deployed on one blockchain (as in the smart contract-based ones) can utilize messaging protocols to offer services to dApps living on another blockchain. In both scenarios, an oracle can offer the same capabilities to any of these blockchains. The use of messaging requires an additional request/response to relay the response to the dApp, so either the request/response or publish-subscribe patterns can be used.

We found that blockchain messaging is used primarily by app-chain oracles; Pyth, Witnet, and Band. Pyth utilizes Wormhole [73], and Witnet uses an internally developed protocol, whereas Band utilizes IBC [70]. As a result, each oracle may support a varying set of blockchains based on the communication capability of such protocols, i.e., number of blockchains they support. In particular, Witnet supports 28 blockchains, Band supports 21 blockchains, while Pyth supports more than 90 blockchains. Supra, which also operates as a general-purpose blockchain, additionally utilizes an internally developed inter-blockchain messaging (supporting 58 blockchains) for oracle data first posted to its layer-1 and desired elsewhere (their oracle has the same name of their network, Supra).

The inter-blockchain messaging approach leads to design-complexity and communication trade-off. It simplifies multi-blockchain support as an oracle needs to deploy one instance of its components, and leaves interacting with various blockchains to the messaging protocol. However, it limits the oracle to what this protocol can offer, and its communication mechanics could in turn be complex.

**Gap 5.** *There is yet no standard inter-blockchain messaging protocol, thus, the protocol choice impacts what blockchains an oracle can service.*

**Multi-instance deployment.** Here, an oracle is replicated across numerous blockchains. Each deployment acts as a separate oracle instance, requiring separate OISC and often separate data feeders. We find that 12 oracles employ this paradigm (Table 4). It can offer more flexibility to data feeders; instead of requiring compatibility with all blockchains supported by an oracle (especially if a data feeder posts responses directly to that blockchain, at least it needs an account on that blockchain to issue transactions), a data feeder can choose to operate on a subset of them. In turn, this would lead to heterogeneous oracle capabilities across these blockchains.

Another discrepancy is related to oracles operating their own tokens. The full functionality of managing this token is typically tied to a single oracle instance. Chainlink, Tellor, API3, UMA, DOS, Redstone and DIAs' tokens are tied to their deployments on Ethereum, whereas Switchboard is tied to its deployment on Solana. See Table 3 for a corresponding list of tokens of each oracle. Thus for oracles operating their own tokens with separate implementations may require asset bridges to be used to manage equivalent token representations across chains. In effect this adds additional design complexity utilizing an oracle supported blockchain where the original token instance does not live, and can lead the incentivization mechanism may vary across oracle instances.

**Insight 12.** *Inter-blockchain messaging may introduce additional security issues for oracle design, e.g., an attack on Wormhole in 2022 caused losing over 300 million USD [93]. While multi-instance deployment leads to silos, magnifying blockchain interoperability problems.*

TABLE 5. ORACLE CLASSIFICATION BASED ON FEE AND REWARD MODELS—WE EXCLUDE TOWN CRIER, DECO, DIOR-SGX, STORK, ZKPASS, AND TLSNOTARY AS REWARDS AND FEES ARE NOT SPECIFIED. UB = NATIVE ASSET OF UNDERLYING BLOCKCHAIN.

| **Oracle** | **Fees and Reward** | | |
|---|---|---|---|
| | Fee asset | Fee type | Reward Asset |
| Astraea | NS | Variable | N/S |
| DAON | NS | Variable | N/S |
| Provable | UB | N/S | UB |
| Chainlink | LINK | Variable | LINK |
| Pyth | UB | Fixed | PYTH |
| Band | BAND | Fixed | BAND |
| API3 | UB | Variable | API3 |
| UMA | Any ERC-20 | Fixed | Any ERC-20 |
| Tellor | TRB | Variable | TRB |
| Witnet | UB | Variable | WIT |
| DOS | N/S | Fixed | DOS |
| DIA | UB | NS | DIA |
| Orcfax | FACT | Variable | FACT |
| Charli3 | N/S | Fixed | C3 |
| Gora | GORA | Variable | GORA |
| Switchoard | SWTCH | Variable | SWTCH |
| Supra | SUPRA | NS | SUPRA |
| Redstone | UB | Variable | UB |

## 6.3. Oracle Fees and Rewards

Oracles, at their core, are financially driven services, relaying external data on-chain in exchange for fees. Considerable effort has been put into the economics of oracle systems specifying the reward structure for data feeders and the oracle fees paid by dApps.

**Oracle fees.** dApps usually pay for the queries they make (separate from the gas fees paid for miners as reported in Appendix A). Fees may vary based on the query complexity and/or the interaction pattern an oracle employs. Namely, immediate-reads do not require fees (or gas) as they are read-only operations. Still an oracle must be compensated for offering such regularly-updated data feeds. In practice, this comes down to external financial agreements between oracles and entities/dApps. For example, for price feeds of assets, these entities could be either those operating the asset or the dApps interested in the price feeds. Usually these external agreements are not public and exact pricing specs are often not directly available. On the other hand, request-response and publish-subscribe patterns require fees at time of request, where for the former, the dApp deposits the total excepted payment of a subscription in an oracle-operated escrow.

Several oracles adopt varying fee hierarchies tied to query type (denoted with "variable" in Table 5). Even these could vary based on the desired security level; in Provable, various authenticity proofs have different fees. Tellor utilizes an auction-based approach, enabling dApps to set their own price via a bid. In other instances, while a traditional fee-hierarchy is not introduced, other requirements are necessary. Chainlink requires the requester to have a certain deposited balance to be able to make arbitrary and computation requests with input privacy—though the cost of request does not change (this deposit is required as a way to deter spam private requests). The remaining examined solutions offer simplified fee models—in what might be considered as an approach to garner more customers. These usually are fixed fees per request or over some period of time; 5 oracles adopt this approach as shown in Table 5.

Typically, fees are either paid in the native token of the blockchain where a dApp lives, or in the oracle-operated token (Table 5). UMA, on the other hand, allows dApps to pay fees in any ERC-20 token. The use of non-native tokens for fees add design complexity as dApps have to additionally obtain and manage the fee asset.

**Reward model.** Data feeders are paid for correctly-fulfilled queries, typically in the oracle-operated token. This includes both oracle-operated tokens (e.g., ERC-20 tokens), and in the case of app-chains digital assets underlying the oracle blockchain. In general, rewards come from the fees paid by dApps when making queries. We find the reward models can be split into three categories.

*Take all.* In centralized models, where an oracle operates the data feeders, the oracle takes all the rewards.

*Split.* Decentralized oracles, on the other hand, have a more complicated payment process, as often rewards are divided between the data feeders and the oracle itself. (We were unable to find exact details regarding reward distribution rates between data feeders and oracles.)

*Staking rewards.* This model encompasses periodical staking rewards (in the oracle-operated token) paid out to data feeders for correct operation—as is the case of app-chains like Pyth, Witnet, and Band. The exact reward specifications may change over time and depend on specific tokenonomic features of the system (e.g., inflationary/fixed asset reserve, disbursement policies, etc.), and the feeder's stake amount.

Another relevant issue is the relationship between assets used for fees and rewards. Reward distribution becomes challenging when fee and reward assets are different. In oracle app-chains this is generally the case—fees are in the native currency of the dApp blockchain, while rewards are distributed in the app-chain token. Thus, additional mechanisms are needed, such as asset bridging and exchanges, to convert between the assets since generally part of the fees are used to pay the feeder's rewards.

**Gap 6.** *Overall, across the examined oracles, little public analysis exist on the rationale behind adopted fee hierarchies/feeder rewards and their effectiveness in oracle operation sustainability.*

## 7. Oracle Capabilities in Practice

We take a closer look at industrial oracles examining their operation in practice. In particular, we conduct empirical evaluations to answer the following questions:

- How are oracles being used today? What is the prevalence of oracle queries and for what purposes?
- How mature are industrial oracle deployments in terms of actual capability support and ease-of-us (compared to their design documentations)?
- How does performance (i.e., query overhead and response freshness/delay) vary across different oracles, and across different blockchains for the same oracle?

We conduct experiments across two axes: oracle usage in practice and capability testing. These allowed us to make observations related to support, use cases (dApp types that use deployed oracles), and oracles' inner-workings regarding how easy (or hard) is to use/test them. We also evaluate how oracles that operate on multiple blockchains compare across these chains. In doing so, the scope of our empirical evaluation covers 16 industrial oracles (those that are active and have mature deployments), and popular blockchains including Ethereum, Algorand, Cardano, Solana, and Ethereum's layer-2 solution Polygon. In this section, we provide an overview of the key findings, while full details of our methodology and experiments/results can be found in Appendix A.

**Oracle use in practice.** We examine oracle usage in practice by collecting on-chain data, as well as off-chain data (e.g., announced partnerships and oracle websites). In terms of number/types of dApps secured by oracles, we find over (mostly DeFi related) 959 dApps, with Chainlink securing around 423 of them.

We also examine query transaction volume on these chains for immediate-read feeds, arbitrary queries, and computation requests. Our findings showcase over $16,983$ transactions on Ethereum and $696,036$ on Polygon for immediate-read feeds, and $4,578$ on Ethereum and 3630 on Polygon for arbitrary data and off-chain computation requests. Among immediate-read feeds, we find Chainlink dominates this volume with $11,779$ transactions on Ethereum and $695,579$ on Polygon. Being a layer-2 solution, we believe that Polygon's lower transaction cost contributed in the high volume of oracle usage on its platform. For Cardano, most of oracle transaction volume is coming from Charli3. Whereas on Solana, only a single oracle was found issuing immediate-read data feeds: Redstone.

**Oracle capabilities.** We examine oracles' capabilities in practice by testing them on the supported blockchain testnets. These tests provide insights on how the actual testing can be done, and what is supported, as well as oracle performance in terms of gas cost, response delay, and ease of use. We create and deploy a series of custom smart contracts on both Ethereum and Polygon testnets making arbitrary URLs and off-chain computation queries.

We test requests from pre-defined datasets—successfully retrieving financial data—and requests to arbitrary sources and off-chain computations—fulfilling retrieval and computation requests on weather data. These covered UMA and Chainlink. For arbitrary queries via Chainlink, the cost was $451,710$ gas units on Ethereum Sepolia versus $279,075$ units (similar costs also found on Polygon Amoy). For computation requests, namely computing the lowest temperature averaged over a month period among two U.S. cities, we find UMA costing $453,936$ gas units, and Chainlink $303,387$ units on Sepolia, with again similar findings on Amoy.

For requests from pre-defined datasets (for the ETH/BTC price pair), these covered Pyth, Stork, Supra, and Tellor. Stork offered the best performance, on Amoy costing $93,861$ gas units, compared to $247,092$ units for Pyth, $386,762$ units for Tellor, and $388,325$ units for Supra. Among these, we found that Stork, Pyth and Supra had zero response delay, all being pull-based oracles, compared to Tellor which took an average of 10 min to return the data request. Finally, we examined privacy support, finding input privacy is currently supported by only one oracle, Chainlink, where secret/private information can be either self-hosted or sent to the DON. We find the DON-hosted costing $295,024$ units of gas on Sepolia compared to $298,723$ units for the self-hosted option.

Overall, we found all oracles offer faster fulfillment on Polygon Amoy than Ethereum Sepolia across all of our experiments.

**Observations.** Throughput our experiments, we made several observations. *Testing was limited for Solana and unviable for Algorand*; in terms of oracle usage, only a single oracle, Redstone on Solana, found to be active, while none were active on Algorand. For testing oracle capabilities, lack of documentations did allow us to test Gora on Algorand testnet, while for Solana, a recent breakage change, that is still being fixed by its team, prevented the testing. Even for Ethereum and Polygon, *many oracles did not offer any testing capabilities*. Also, some blockchains have very limited support; Cardano only supports immediate-read data feeds.

Moreover, *lack of clear and up-to-date documentation* was an on-going problem across our examinations. In many occasions, interaction with the development teams of the oracles was necessary to resolve some of the challenges and obtain permission to deploy arbitrary queries. These reflect on the usability of oracles and may limit which aspects can be tested. Furthermore, oracle privacy is still in its infancy, *only input privacy is offered and by only one oracle—Chainlink*; this points to the greater need for further privacy consideration and support.

Overall, a recurring issue that we observed is that *testnets do not support all oracle capabilities available on the mainnet*. In fact, only four oracles, UMA, Pyth, Supra, and Chainlink, offer testing over both Polygon and Ethereum testnets (and in case of UMA, no data feeders were active, forcing us to self-propose query responses). Given testnets' vital role in the development process, capabilities offered on testnets play a crucial role in dApp secure development cycle; without this, dApps that use oracles are forced to integrate directly into the mainnet without full testing. Aside from the additional cost and scalability implications of testing, this may lead to security issues (smart contract bugs resulting in millions of dollars being lost [94] only further emphasize this point).

## 8. Concluding Remarks and the Road Ahead

Today, oracles represent a critical piece of blockchain infrastructure. Existing solutions encompass a wide assortment of individual designs under varying goals and assumptions, which makes it hard to comprehend this landscape and build on top of it. Our work aims to bridge this gap; it provides a holistic and granular analysis of current oracle solutions, it identifies the gaps and open problems, and it offers a guide for practitioners on which oracles to use when building applications.

We conclude with highlighting potential future work directions based on the gaps we identify, in addition to oracle implications in terms of connecting blockchains to legacy systems, relation to MEV, potential misuses, and their interplay with AI.

### 8.1. Towards a Security Notion for Oracles

One of the notable gaps we identified is the lack of a generic security notion for oracles. Oracles have a complicated functionality that involves several components and parties interacting with each other. Thus, defining their security requires a framework that can captures these dynamics, as well as how oracles are composed with dApps and the underlying blockchain modules. Thus, a framework like the Universal Composability (UC) [95], or the Interactive Universal Composability (iUC) framework [96] that prior work [97] argued to be better for blockchain-based protocols than standard UC, are more suitable for modeling oracle security than property-based definitions in the stand alone model.

Furthermore, decentralized oracles that employ an app-chain (which is a full blockchain) or a DON, require covering the additional security requirements or properties of these components, including liveness and safety. This only attests to the importance of following a composable approach that produces a modular security notion. Moreover, as incentives play a crucial role in oracles, whether it is on the reward side or the financial penalty side, these economical factors must be considered in the security notion. In other words, the developed notion should capture capture financial-based security, by having game theoretic/incentive compatibility components, as well as the conventional security guarantees.

## 8.2. Expressive Privacy Support

Current privacy support in oracles is limited to I/O privacy by encrypting these values such that only the DON or a TEE can decrypt. Users may not trust the DON to reveal nothing about their data. Also, TEEs come with the assumption of resisting side channel attacks and reverse engineering which has been challenged, thus it is desirable to minimize the tasks assigned to a TEE.

Oracles can be utilized to achieve more sophisticated privacy capabilities. In particular, due to scalability issues of blockchains, it is undesirable to run heavy cryptographic machinery on-chain. Instead, these can be executed off-chain with the oracle reporting the result for on-chain use. For example, if a dApp requires some computation over private data held by multiple parties, those parties can run an MPC protocol off-chain and the oracle reports the computation result. Another example is having a ZKP being verified by a TEE, and only relay the TEE's attestation that the proof is valid (so no expensive on-chain proof verification). Even proof generation can utilize the pull-based oracle architecture. That is, have the oracle network offer a delegated ZKP generation service using, e.g., distributed delegated proving [98], [99] or collaborative ZKPs [100]. Then, the user submits the proof for verification (either on-chain or for a TEE as above).

There have been some privacy-preserving blockchains in this vein, e.g., Partisia [101] supports off-chain MPC and Oasis Sapphire [102] utilizes TEEs to execute off-chain private computations. However, we argue that doing that via oracles improves interoperability and enables general-purpose blockchain support.

## 8.3. Misuses of Oracles

Unfortunately, it has been shown that oracle-based smart contracts can be used for malicious purposes under the notion of criminal smart contracts (CSCs) [103], [104]. In particular, smart contracts and oracles are utilized to orchestrate attacks in a decentralized/trustless way, e.g., stealing private information, distributed denial of service attacks (DDoS), bribery, or even harming individuals. As the attack targets reside outside the blockchain ecosystem, the smart contract relies on oracles to report the attack result, and if the attack is found to be successful, attackers who registered in this contract obtain their rewards. For example, success of DDoS can be reported by measuring the delay of a URL query made by the oracle to a particular website, or checking that a certain (offensive) event happened can be reported by the oracle via a query to, e.g., major news websites.

Defending against CSCs has been briefly discussed in prior work, with the argument that miners should vet contracts and reject the deployment of those with malicious goals. However, attackers may craft these contracts in a way that make them look benign during the deployment phase. This highlights an important open question related to vetting these contracts after they are deployed, based on their oracle interactions and transaction flow to discover any malicious behaviors.

## 8.4. Maximal Extractable Value (MEV)

MEV has received increased interest lately, with a myriad of works studying its strategies, mitigation, and fair distribution of extracted value (when it cannot be prevented). Oracle extractable value (OEV) is a recent notion capturing MEV resulted from oracle interaction. OEV arises when an oracle update (i.e., query fulfillment) triggers downstream state changes to dApps that may result in financial gains. OEV strategies rely on reordering/-managing oracle-related transactions. Understanding OEV is an on-going question, with recent industry reports [105], [106] exploring both its strategies and scope.

**OEV strategies.** These fall under three categories:

*Front/back running.* These rely on monitoring the mempool for pending oracle updates, and then inserting a value-extracting transaction(s) to occur immediately before or after the update, e.g., front-run a price feed update to execute a trade at a more favorable price than the new one. Immediate-read oracles, where data is updated periodically, is particularly susceptible to this attack. On the other hand, pull-based oracles—that combine data fulfillment into the same transaction of a dApp operation—can mitigate that as they eliminate the update window.

*Arbitrage.* This strategy takes advantage of price inconsistencies between various oracle-integrated dApps, usually within small periods when price data on at least one dApp, within the same blockchain or across different blockchains, is misaligned with the true market price. In the context of oracles, discrepancies of price feeds offered by several oracles within the same blockchain, or even same oracle across different blockchains, may present arbitrage opportunities. Arbitrage is not inherently adversarial, as it is a vital component of market infrastructure, but it is still considered a value-extraction strategy.

*Liquidation.* Liquidation can be viewed as OEV-specific strategy (rather than a generic MEV one); it is tied to the inherent relationship of oracle updates carrying price data to be used to balance lending protocols like AAVE [8], Compound [9], and Maker [107]. Liquidation consists of monitoring lending protocols for soon-to-be liquidations and inserting a transaction liquidating a position as soon as this position becomes insolvent. In detail, when an oracle update triggers a collateral liquidation threshold in a lending protocol, a liquidator can come in to repay the borrower's debt and claim their collateral, thus taking the delta as profit. Similar to arbitrage, liquidation is not inherently an adversarial strategy, as it is critical component of lending protocol's overall solvency. However, it still represents a value-extraction strategy since transaction inclusion order determines who can liquidate the position and collect the profits.

**O/MEV redistribution and mitigation techniques.** We briefly discuss some of the techniques that aim to mitigate MEV or redistribute the extracted value back to the actors in the network. These solutions can be split into: blockchain level redistribution, application-specific redistribution, and privacy-enhancing tooling. We note that all these techniques are for generalized MEV, but indeed can be employed for OEV; our discussion of application-specific redistribution techniques is tailored to OEV.

*Blockchain-level redistribution* The introduction of decentralized block building markets [108] has reshaped transaction settlement. They separate the party roles into a searcher (the party identifying value-extracting strategies), a block-builder (who aggregates transactions into candidate blocks) and a proposer (a validator with proposal rights). Protocols such as MEV-Boost [109] and MEV-blocker [110] adopt this design to redirect profits from searchers and block-builders to proposers while improving transaction fairness. Moreover, protocols like MEV-Share [111] extend this mission to redistribute a fraction of the collected MEV back to the originating users.

*Application-specific redistribution.* Liquidation-based OEV is intrinsic to lending protocols. While blockchain-level solutions remain valuable, lending protocols can incorporate internal auctions for liquidation rights capturing value to be redistributed to involved actors (i.e., lending protocol participants, validators, and block builders). To date, UMA's Oval [112], Pyth's express relay [113], and API3's OEV Network [114] have implemented such auctions where they return the (winning) bid back to the lending protocol. These efforts call for application-specific redistribution mechanisms where value extraction is managed at the protocol level rather than being delegated to a global MEV infrastructure.

*Privacy-enhancing mechanisms.* The main reason behind MEV is having public blockchains that enable anyone to see published/pending transactions. Thus, removing this visibility, by having private transactions, mitigates MEV. Privacy-based techniques have various flavors. Private order flow networks route pending transactions to a network of block-builders who build blocks privately (in practice those builders employ TEEs such as within BuilderNet [115]). Private RPC endpoints, like MEVBlocker [110] and Flashbot's Protect [116], have users send transactions directly to a single trusted block-builder to avoid exposure to the public mempool. Finally, batched threshold encryption techniques [117], [118] keep transactions encrypted until a block is finalized, then it is decrypted. These flavors have different tradeoffs between privacy, security, efficiency, and economic interest.

**Discussion.** Studying OEV is still lacking. Current efforts have focused on generalized MEV within specific blockchains, such as Ethereum. Oracles, especially those operating app-chains and employing blockchain messaging protocols, introduce new layers of OEV. Examples include additional front- and back-running opportunities tied to the use of cross-chain messaging, as well as new arbitrage positions tied to differences in messaging latency between blockchains.

Beside studying the impact of specific strategies, another interesting research avenue is exploring domain-specific OEV redistribution solutions. For example, developing oracle priority lanes—separate designation and inclusion policies for oracle transaction within a block, or oracle-aware block building markets that specifically account for time-sensitive oracle updates while allowing finer-grain control and redistribution policies. A key challenge here is balancing fairness, decentralization, and effective OEV-recapture while ensuring that such mechanisms do not introduce new OEV vectors.

In terms of mitigation, it is unknown how privacy-enhancing systems will mature within decentralized block building markets and the broader blockchain ecosystem. More work is needed to design generic systems that provide flexible privacy, e.g., Flashbot's envisioned SUAVE [119] that outlines a holistic system for programmable-privacy. Another direction is exploring partial privacy, e.g., employ ZKPs so that users reveal certain properties of a transaction while achieving the same fairness guarantees that one would gain if transactions are fully private. Nonetheless, the additional cost of privacy may deter adoption, which highlights the importance of devising efficient solutions and alternative fair revenue redistribution mechanisms.

## 8.5. Partnerships with Legacy Systems

The growing interest in blockchain oracles, and recognizing their value in expanding the application classes of blockchains, have driven increasing engagement from legacy institutions. This is through not only the use of oracle services and blockchain integrations, but also the release of previously siloed data for on-chain use.

Financial service providers [120], [121], major new agencies [122], and even government agencies [123], [124] have begun publishing authenticated data on-chain via oracles. Concurrently, a number of proof-of-concept initiatives are demonstrating how oracles and blockchain technology can reshape the data infrastructure of traditional institutions. For instance, a recent collaboration between Chainlink and a consortium of 24 major financial institutions and market infrastructure providers [125], including Swift, DTCC, UBS, and DBS, showcased the use of oracles for standardizing the extraction and delivery of corporate action data while satisfying corporate compliance standards such as ISO 27001 and SOC 2 [126]. Similar pilots with Revolut [121], Sygum Bank and Fidelity International [120], show how fund's Net Asset Value (NAV) and pricing data can be distributed on-chain to support digital asset markets.

Looking ahead, the deepening collaboration between oracle systems and legacy institutions signals a growing maturity of corporate use of blockchains, and a broader shift that will shape the future of on-chain data availability, digital market regulation, and institutional accountability.

## 8.6. Oracles and AI

The recent explosion of high-quality LLMs led to the notion of AI agents. These agents have sophisticated reasoning capabilities, are able to understand and generate natural languages, and are very effective in automating complex tasks in data analysis and interpretation. A natural question that attracted the attention of the community is whether AI can be combined with blockchains. Some of these efforts viewed oracles as a vital piece by acting as a proxy that connects an AI agent to the blockchain to achieve several goals. Furthermore, many works have explored the utility of ML-based techniques in improving oracle reliability and trustworthiness. We elaborate on both of these directions below.

**ML-based anomaly and manipulation detection.** In the context of oracles, ML-based uncertainty prediction can be used to identify outliers, coordinated manipulation attempts, and sudden deviation in oracle responses—either across multiple oracles or across diverse data sources within a single oracle. A recent survey [127] has explored this space, discussing how such ML-assisted models tend to improve existing statistical aggregation approaches in both benign [128] (preventing inaccurate or noisy data being sent downstream) and malicious ones [129], [130] (ensuring that coordinated manipulation attempts that occur upstream, from either dishonest data providers or data feeders, do not propagate downstream).

Such approaches do not change the underlying nature of blockchain oracles, instead they provide an additional protection layer ensuring fail-safes for dApps. Also, they only flag anomalies in data before it is propagated downstream to dApps. Yet, these dApps still need data, requiring fallback mechanisms or facing downtime. An area of interest is designing decentralized fallback mechanisms that suit different oracle designs without adding new security concerns to the table.

**LLMs and Agentic AI.** Integrating AI agents with blockchains [131] can range from a "conversation-agent" that retrieves/reports on-chain data, to more advanced "goal-centric" agents capable of issuing queries and submitting transactions on-chain according to some predefined goal, e.g., portfolio optimization. Within oracles, this integration can take the form of an oracle serving as a proxy for an LLM, allowing dApps to obtains answers to natural language queries, or a hybrid model where an LLM complements conventional data source queries by verifying and contextualizing retrieved data (i.e., fact extraction and verification [127]). On the flip side, oracles and blockchains can be used as a complementary tool to LLM [132], offering real-time data to a pre-trained model, which represents a decentralized blockchain-based Retrieval-Augmented Generation (RAG).

Relevant efforts are being deployed in practice. Some oracles have begun preparing for AI commerce, i.e., streamlining processes for autonomous agent-to-agent interaction. Switchboard has announced a partnership with Corbit SDK [133] (who incorporates the x402 payment standard [134] into conventional APIs), providing on-chain payment rails for agents, which in turn would allow AI agents to autonomously make on-chain oracle queries. Early prototypes of LLM-powered oracles include pilots where multiple LLMs are tasked to parse data such as corporate data [135], and resolve ambiguous real-world events for prediction markets [136].[7] Also, Supra has released a whitepaper [137] discussing the concept of multi-agent AI oracles (i.e., aggregation from multiple LLMs) that is currently under development.

Despite offering expanding capabilities to blockchain oracles, AI agents do not solve the oracle problem. Instead, they face the same foundation concerns that underlie the oracle problem itself. That is, data generated by AI agents remain non-deterministic, prone to hallucination [138], and prompt bias [139]–[141], reintroducing the core oracle problem of determining an agreed-upon truth. Aggregation and consensus based approaches can be utilized among multiple agents,[8] but the additional overhead and lack of cryptographic verifiability without additional off-chain infrastructure [127] add new security concerns. This compounds with the security and privacy considerations that arise within LLMs, with attacks including prompt injections, data poisoning, and prompt leakage [142]–[144]. Overall, it is clear that AI will be pivotal in expanding the capabilities of blockchain oracles. Nonetheless, more work is needed to understand the design implications of AI agents as part of the oracle system as well as additional security, privacy, and complexity considerations that come with LLM-based fact extraction and verification within these systems.

## Acknowledgments

This work is supported by NSF under grant No. CNS-2226932.

## References

[1] L. Argento, F. Buccafurri, A. Furfaro, S. Graziano, A. Guzzo, G. Lax, F. Pasqua, and D. Saccà, "Id-service: A blockchain-based platform to support digital-identity-aware service accountability," *Applied Sciences*, vol. 11, no. 1, 2020.

[2] "Fractal id - web3 identity solution provider," https://web.fractal.id/.

[3] "Axie infinity," https://axieinfinity.com/.

[4] J. Proelss, S. Sévigny, and D. Schweizer, "Gamefi: The perfect symbiosis of blockchain, tokens, defi, and nfts?" *International Review of Financial Analysis*, vol. 90, 2023.

[5] S. Werner, D. Perez, L. Gudgeon, A. Klages-Mundt, D. Harz, and W. Knottenbelt, "Sok: Decentralized finance (defi)," in *ACM Conference on Advances in Financial Technologies*, 2022.

[6] "Dydx," https://dydx.exchange/.

[7] "Synthetix," https://synthetix.io/.

[8] "Aave," https://aave.com/.

7. This has been used in Polymarket [60], where they found that the LLM-determined outcome matched that of the real-world truth value across various event categories 89.3% of the time, where among, sporting related events this increased to 99.7% of the time.

8. While consensus may be enough to reach an agreement, the use of datasets shared among multiple agents may introduce biases within consensus.

[9] R. Leshner and G. Hayes, "Compound: The money market protocol," 2019, https://compound.finance/documents/Compound.Whitepaper.pdf.

[10] "Authena," https://authena.io/.

[11] "Dimitra,," https://dimitra.io/.

[12] DefiLlama, "Oracles - oracle rankings," https://defillama.com/oracles.

[13] "Tlsnotary - a mechanism for independently audited https sessions," https://tlsnotary.org/TLSNotary.pdf.

[14] "Truthcoin - peer-to-peer oracle system and prediction marketplace," 2015, https://www.truthcoin.info/papers/truthcoin-whitepaper.pdf.

[15] J. Teutsch and C. Reitwießner, "A scalable verification solution for blockchains," in *Aspects of Computation and Automata Theory with Applications*, 2024.

[16] "Provable (formely Oraclize)," https://app.provable.xyz/home/features.

[17] "Chainlink," https://chain.link/.

[18] "Tellor: A decentralized oracle," https://docs.tellor.io/tellor/whitepaper/introduction.

[19] "UMA data verification mechanism: Adding economic guarantees to blockchain oracles," https://github.com/UMAprotocol/whitepaper/blob/master/UMA-DVM-oracle-whitepaper.pdf.

[20] V. Buterin, "Schellingcoin: A minimal-trust universal data feed,," 2014, https://blog.ethereum.org/2014/03/28/schellingcoin-a-minimal-trust-universal\-data-feed.

[21] J. Adler, R. Berryhill, A. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A decentralized blockchain oracle," in *2018 IEEE international conference on internet of things (iThings) and IEEE green computing and communications (GreenCom) and IEEE cyber, physical and social computing (CPSCom) and IEEE smart data (SmartData)*, 2018.

[22] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in *ACM SIGSAC conference on computer and communications security*, 2016.

[23] F. Zhang, D. Maram, H. Malvai, S. Goldfeder, and A. Juels, "Deco: Liberating web data using decentralized oracles for tls," in *ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[24] "Maker Protocol - Oracle Module," https://docs.makerdao.com/smart-contract-modules/oracle-module.

[25] "Chronicle protocol - cost-efficient, verifiable blockchain oracles," https://chroniclelabs.org/.

[26] "Zkpass," https://zkpass.org/#about.

[27] "DIA," https://www.diadata.org/.

[28] S. Woo, J. Song, and S. Park, "A distributed oracle using intel sgx for blockchain-based iot applications," *Sensors*, vol. 20, no. 9, 2020.

[29] "Band protocol," https://bandprotocol.com/.

[30] "Pyth network," https://pyth.network/.

[31] "TLSNotary," https://tlsnotary.org/.

[32] R. Kamiya, "Shintaku: An end-to-end-decentralized general-purpose blockchain oracle system," 2018, https://gitlab.com/shintaku-group/paper/raw/master/shintaku.pdf.

[33] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, "Augur: a decentralized oracle and prediction market platform," *arXiv preprint arXiv:1501.01042*, 2015.

[34] "DOS Network," https://dos.network/.

[35] "API3: Decentralized APIs for Web 3.0," https://docs.api3.org/api3-whitepaper-v1.0.3.pdf.

[36] "Witnet," https://witnet.io/.

[37] "Redstone oracles," https://redstone.finance/.

[38] "Switchboard," https://switchboard.xyz/.

[39] "Supra," https://supra.com/.

[40] "Stork," https://www.stork.network/.

[41] A. Beniiche, "A study of blockchain oracles," *arXiv preprint arXiv:2004.07140*, 2020.

[42] H. Al-Breiki, M. H. U. Rehman, K. Salah, and D. Svetinovic, "Trustworthy blockchain oracles: review, comparison, and open research challenges," *IEEE access*, vol. 8, 2020.

[43] A. Pasdar, Y. C. Lee, and Z. Dong, "Connect api with blockchain: A survey on blockchain oracle implementation," *ACM Computing Surveys*, vol. 55, no. 10, 2023.

[44] Y. Zhao, X. Kang, T. Li, C.-K. Chu, and H. Wang, "Toward trustworthy defi oracles: past, present, and future," *IEEE Access*, vol. 10, 2022.

[45] S. K. Ezzat, Y. N. Saleh, and A. A. Abdel-Hamid, "Blockchain oracles: State-of-the-art and research directions," *IEEE Access*, vol. 10, 2022.

[46] S. Eskandari, M. Salehi, W. C. Gu, and J. Clark, "Sok: Oracles from the ground truth to market manipulation," in *ACM Conference on Advances in Financial Technologies*, 2021.

[47] "Uniswap," https://app.uniswap.org/.

[48] B. Liu, P. Szalachowski, and J. Zhou, "A first look into defi oracles," in *IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS)*, 2021.

[49] R. Gansäuer, H. B. Aoun, J. Droll, and H. Hartenstein, "Price oracle accuracy across blockchains: A measurement and analysis," in *International Workshop on Cryptoasset Analytics (CAAW)*, 2025.

[50] A. Gangwal, R. Valluri, and M. Conti, "Analyzing price deviations in defi oracles," in *International Conference on Cryptology and Network Security*, 2022.

[51] L. W. Cong, E. S. Prasad, and D. Rabetti, "Financial and informational integration through oracle networks," National Bureau of Economic Research, Tech. Rep., 2025.

[52] J. Dong, C. Song, Y. Sun, and T. Zhang, "Daon: A decentralized autonomous oracle network to provide secure data for smart contracts," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 5920–5935, 2023.

[53] M. Merlini, N. Veira, R. Berryhill, and A. Veneris, "On public decentralized ledger oracles via a paired-question protocol," in *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019.

[54] Y. Cai, N. Irtija, E. E. Tsiropoulou, and A. Veneris, "Truthful decentralized blockchain oracles," *International Journal of Network Management*, vol. 32, no. 2, 2022.

[55] K. Nelaturu, J. Adler, M. Merlini, R. Berryhill, N. Veira, Z. Poulos, and A. Veneris, "On public crowdsource-based mechanisms for a decentralized blockchain oracle," *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1444–1458, 2020.

[56] "Orcfax," https://orcfax.io/.

[57] "Charli3," https://charli3.io/.

[58] "Gora," https://www.gora.io/.

[59] "Edge protocol," https://chaoslabs.xyz/edge.

[60] "Polymarket," https://polymarket.com/.

[61] "IPFS," https://www.ipfs.com/.

[62] J. Teutsch, F. Kattan, and B. Sims, "Truebit unchained a technical report on transparency," https://truebit.io/wp-content/uploads/2024/03/Truebit-Unchained-Technical-Report-1.pdf.

[63] H. Kalodner, S. Goldfeder, X. Chen, S. M. Weinberg, and E. W. Felten, "Arbitrum: Scalable, private smart contracts," in *USENIX Security*, 2018.

[64] "Optimism," https://optimism.io/.

[65] T. Liu, T. Xie, J. Zhang, D. Song, and Y. Zhang, "Pianist: Scalable zkrollups via fully distributed zero-knowledge proofs," in *IEEE Symposium on Security and Privacy*, 2024.

[66] "ZKsync," https://zksync.io/.

[67] Introducing sp1-cc: Unleashing the full power of the evm. [Online]. Available: https://blog.succinct.xyz/sp1-cc/

[68] "Sei blockchain," https://www.sei.io/.

[69] "Unichain," https://www.unichain.org/.

[70] "IBC Protocol," https://ibcprotocol.org/.

[71] "Charli3 pull oracle," https://charli3.io/products/pull-oracle.

[72] "Chainlink data stream launches," urlhttps://www.prnewswire.com/news-releases/chainlink-data-streams-launches-on-mainnet-as-new-high-speed-oracle-solution-for-defi-301944062.html.

[73] "Wormhole," https://wormhole.com/.

[74] "How Pyth Works - Hermes," urlhttps://docs.pyth.network/price-feeds/how-pyth-works/hermes.

[75] J. Garay, A. Kiayias, and N. Leonardos, "The bitcoin backbone protocol: Analysis and applications," in *EUROCRYPT*, 2015.

[76] R. Pass, L. Seeman, and A. Shelat, "Analysis of the blockchain protocol in asynchronous networks," in *EUROCRYPT*, 2017.

[77] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A.-R. Sadeghi, "Software grand exposure:{SGX} cache attacks are practical," in *USENIX workshop on offensive technologies (WOOT)*, 2017.

[78] J. Van Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx, "Foreshadow: Extracting the keys to the intel {SGX} kingdom with transient {Out-of-Order} execution," in *USENIX Security Symposium*, 2018.

[79] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller, "Cache attacks on intel sgx," in *European Workshop on Systems Security*, 2017.

[80] G. Chen, S. Chen, Y. Xiao, Y. Zhang, Z. Lin, and T. H. Lai, "Sgxpectre: Stealing intel secrets from sgx enclaves via speculative execution," in *IEEE European Symposium on Security and Privacy (EuroS&P)*, 2019.

[81] I. Puddu, M. Schneider, M. Haller, and S. Čapkun, "Frontal attack: Leaking {Control-Flow} in {SGX} via the {CPU} frontend," in *USENIX Security Symposium*, 2021.

[82] J. Chuang, A. Seto, N. Berrios, S. van Schaik, C. Garman, and D. Genkin, "Tee.fail: Breaking trusted execution environments via ddr5 memory bus interposition," in *IEEE Symposium on Security and Privacy*, 2026.

[83] "Eigenlayer," https://www.eigenlayer.xyz/.

[84] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, pp. 1–47, 2017.

[85] "Matic network," https://matic.network/.

[86] P. Chakka, S. Joshi, A. Kate, J. Tobkin, and D. Yang, "Oracle agreement: From an honest super majority to simple majority," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2023.

[87] "Saftey net depreciation," https://developer.android.com/training/safetynet/deprecation-timeline.

[88] "Ledger," https://www.ledger.com/.

[89] C. Weng, K. Yang, J. Katz, and X. Wang, "Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits," in *IEEE Symposium on Security and Privacy*, 2021.

[90] K. Yang, P. Sarkar, C. Weng, and X. Wang, "Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field," in *ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[91] "Tellor layer," https://tellor.io/blog/tellor-layer/.

[92] "Cosmos," https://cosmos.network/.

[93] "Wormhole 325 million dollar hack," https://www.theverge.com/2022/2/3/22916111/wormhole-hack-github-error-325-million\-theft-ethereum-solana.

[94] "197 Million Stolen: Euler Finance Flash Loan Attack Explained," https://www.chainalysis.com/blog/euler-finance-flash-loan-attack/.

[95] R. Canetti, "Universally composable security: A new paradigm for cryptographic protocols," in *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, 2001.

[96] J. Camenisch, S. Krenn, R. Küsters, and D. Rausch, "iuc: Flexible universal composability made simple," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2019.

[97] M. Graf, D. Rausch, V. Ronge, C. Egger, R. Küsters, and D. Schröder, "A security framework for distributed ledgers," in *ACM SIGSAC Conference on Computer and Communications Security*, 2021.

[98] A. Chiesa, R. Lehmkuhl, P. Mishra, and Y. Zhang, "Eos: Efficient private delegation of {zkSNARK} provers," in *USENIX Security Symposium*, 2023.

[99] Y. Yang, Y. Cheng, K. Wang, X. Li, J. Sun, J. Shen, X. Dong, Z. Cao, G. Yang, and R. H. Deng, "Siniel: Distributed privacy-preserving zksnark," in *Network and Distributed System Security Symposium (NDSS)*, 2025.

[100] A. Ozdemir and D. Boneh, "Experimenting with collaborative {zk-SNARKs}:{Zero-Knowledge} proofs for distributed secrets," in *USENIX Security Symposium*, 2022.

[101] "Partisia," https://partisiablockchain.com/.

[102] "Oasis sapphire," https://oasis.net/sapphire.

[103] A. Juels, A. Kosba, and E. Shi, "The ring of gyges: Investigating the future of criminal smart contracts," in *ACM SIGSAC Conference on Computer and Communications Security*, 2016.

[104] Z. Motaqy, G. Almashaqbeh, B. Bahrak, and N. Yazdani, "Bet and attack: Incentive compatible collaborative attacks using smart contracts," in *International Conference on Decision and Game Theory for Security (GameSec)*, 2021.

[105] "Api3: The state of oev," https://members.delphidigital.io/reports/api3-the-state-of-oev.

[106] R. Humphry, P. Avramovic, F. Acevedo, S. Alkhair, and A. Gervais, "Review of maximal extractable value & blockchain oracles," Technical report, Financial Conduct Authority, Tech. Rep., 2024.

[107] "The dai stablecoin system," https://makerdao.com/whitepaper/DaiDec17WP.pdf.

[108] "Proposer–builder separation: Friendly fee market designs," https://ethresear.ch/t/proposer-block-builder-separation-friendly-fee-market-designs/9725/.

[109] "Mev-boost: implementation of proposer-builder separation (pbs) for ethereum," https://boost.flashbots.net/.

[110] "Mevblocker," https://mevblocker.io/.

[111] "Mev-share," https://docs.flashbots.net/flashbots-mev-share/introduction.

[112] "Oval: Oracle value aggregation layer," https://uma.xyz/oval.

[113] "Express relay: Priority auctions for efficient defi markets," https://www.pyth.network/blog/express-relay-priority-auctions.

[114] "Oev network – in depth," https://docs.api3.org/oev-searchers/in-depth/oev-network/.

[115] "Buildernet: A decentralized block-building network for ethereum," https://buildernet.org/.

[116] "Flashbot's protect rpc," https://docs.flashbots.net/flashbots-protect/overview.

[117] A. R. Choudhuri, S. Garg, J. Piet, and G.-V. Policharla, "Mempool privacy via batched threshold encryption: Attacks and defenses," in *USENIX Security Symposium*, 2024.

[118] J. Bormet, S. Faust, H. Othman, and Z. Qu, "{BEAT-MEV}: Epochless approach to batched threshold encryption for {MEV} prevention," in *USENIX Security Symposium*, 2025.

[119] "The future of mev is suave," https://writings.flashbots.net/the-future-of-mev-is-suave.

[120] "Sygnum and fidelity international partner with chainlink to provide fund nav data onchain," https://www.sygnum.com/news/sygnum-and-fidelity-international-partner-with-chainlink-to-\provide-fund-nav-data-onchain/.

[121] “Revolut’s digital asset market data integrated into the Pyth Network,” https://financialit.net/news/blockchain/revoluts-digital-asset-market-data-integrated-pyth-network.

[122] “Ap, chainlink to bring trusted data onto leading blockchains,” https://www.ap.org/media-center/press-releases/2021/ap-chainlink-to-bring-trusted-data-onto-leading-blockchains/.

[123] “The U.S. Department of Commerce and Chainlink are now bringing government macroeconomic data onchain,” https://blog.chain.link/united-states-department-of-commerce-macroeconomic-data/.

[124] “A new wave of economic data is now onchain,” https://www.pyth.network/blog/a-new-wave-of-economic-data-is-now-onchain.

[125] “Chainlink’s work with major banking and capital markets institutions,” https://blog.chain.link/chainlinks-work-with-major-banking-and-capital-\markets-institutions/.

[126] “Chainlink becomes first data and interoperability oracle platform to achieve ISO 27001 and SOC 2 compliance,” https://blog.chain.link/chainlink-achieves-iso-soc-2-certifications/.

[127] G. Caldarelli, “Can artificial intelligence solve the blockchain oracle problem? unpacking the challenges and possibilities,” *arXiv preprint arXiv:2507.02125*, 2025.

[128] S. Park, O. Bastani, and T. Kim, “{ACon^2}: Adaptive conformal consensus for provable blockchain oracles,” in *USENIX Security Symposium*, 2023.

[129] D. P. Isravel, J. Punitha, and M. Dhas, “Reinforcement learning-enhanced adaptive blockchain oracles for secure and efficient data aggregation,” in *Journal of Information Systems Engineering and Management*, 2025.

[130] B. Gao, Y. Wang, Q. Wei, Y. Liu, R. S. M. Goh, and D. Lo, “Airaclex: Automated detection of price oracle manipulations via llm-driven knowledge mining and prompt generation,” *arXiv preprint arXiv:2502.06348*, 2025.

[131] N. Romandini, C. Mazzocca, K. Otsuki, and R. Montanari, “Sok: Security and privacy of ai agents for blockchain,” *arXiv preprint arXiv:2509.07131*, 2025.

[132] S. Fu and M. Xie, “Ai oracle: A blockchain-powered oracle for llms and ai agents,” in *Crypto Valley Conference (CVC)*, 2025.

[133] “Ready your apps for agentic commerce,” https://corbits.dev/.

[134] “x402 - an open protocol for internet-native payments,” https://www.x402.org/.

[135] “Transforming asset servicing with ai, oracles, and blockchains,” https://pages.chain.link/hubfs/e/transforming-asset-servicing.pdf.

[136] “Empirical evidence in ai oracle development,” https://blog.chain.link/ai-oracles/.

[137] “Threshold ai oracles: Verified ai for event-driven web3,” https://supra.com/documents/Threshold_AI_Oracles_Supra.pdf.

[138] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, “Survey of hallucination in natural language generation,” *ACM computing surveys*, vol. 55, pp. 1–38, 2023.

[139] D. Huang, J. M. Zhang, Q. Bu, X. Xie, J. Chen, and H. Cui, “Bias testing and mitigation in llm-based code generation,” *ACM Transactions on Software Engineering and Methodology*, 2024.

[140] Y. Li, M. Du, R. Song, X. Wang, and Y. Wang, “A survey on fairness in large language models,” *arXiv preprint arXiv:2308.10149*, 2023.

[141] S. K. Sakib and A. B. Das, “Challenging fairness: A comprehensive exploration of bias in llm-based recommendations,” in *International Conference on Big Data*, 2024.

[142] Y. Liu, G. Deng, Y. Li, K. Wang, Z. Wang, X. Wang, T. Zhang, Y. Liu, H. Wang, Y. Zheng *et al.*, “Prompt injection attack against llm-integrated applications,” *arXiv preprint arXiv:2306.05499*, 2023.

[143] B. Hui, H. Yuan, N. Gong, P. Burlina, and Y. Cao, “Pleak: Prompt leaking attacks against large language model applications,” in *ACM SIGSAC Conference on Computer and Communications Security*, 2024.

[144] Anthropic, “A small number of samples can poison llms of any size,” https://www.anthropic.com/research/small-samples-poison.

[145] “Polygon,” https://polygon.technology/.

[146] “Etherscan,” https://etherscan.io/.

[147] “Polygonscan,” https://polygonscan.com/.

[148] “Cardanoscan,” https://cardanoscan.io/.

[149] “Introducing the amoy testnet for polygon pos,” https://polygon.technology/blog/introducing-the-amoy-testnet-for-polygon-pos.

[150] “Sepolia,” https://ethereum.org/en/developers/docs/networks/#sepolia.

[151] “Solidity,” https://docs.soliditylang.org/en/latest/.

[152] “Hardhat,” https://hardhat.org/.

[153] “Infura,” https://www.infura.io/.

[154] “Metamask,” https://metamask.io/.

[155] “Chainlink Functions Toolkit,” https://www.npmjs.com/package/@chainlink/functions-toolkit.

[156] “Supra Oracle - REST API,” https://docs.supra.com/oracles/apis-real-time-and-historical-data/rest-api.

[157] “Gora feed explorer,” https://app.gora.io/feeds/prices.

[158] “Redstone finance - push model (on-chain feeds),” https://app.redstone.finance/app/feeds/.

[159] “Chainlink - data feeds,” https://data.chain.link/feeds.

[160] “Supra - Data,” https://supra.com/data.

[161] “Anchor,” https://www.anchor-lang.com/docs.

[162] “Anchor release note: 0.31.1,” https://github.com/solana-foundation/anchor/blob/v0.31.1/docs/content/docs/updates/release-notes/0-31-1.mdx#proc-macro2-build-fix.

[163] “TaskOn Selects ZkPass to enhance KYC Capabilities—Medium,” https://medium.com/@taskonxyz/taskon-selects-zkpass-to-enhance-kyc-capabilities-and-strengthen\-proof-of-humanity-9a17289778a0.

[164] “Orcfax - feed explorer,” https://explorer.orcfax.io/.

[165] “Charli3 - oracle feeds,” https://portal.charli3.io/dev/feeds.

[166] “Witnet data feeds explorer,” https://feeds.witnet.io/.

[167] “Open-meteo: open-source weather api,” https://open-meteo.com/.

[168] “Consensys announces the sunset of truffle and ganache.” https://consensys.io/blog/consensys-announces-the-sunset-of-truffle-and-ganache-and-new\-hardhat.

[169] “Chainlink Functions,” https://functions.chain.link/.

[170] “Github Gist,” https://gist.github.com/.

[171] “Stork Oracle - REST API,” https://docs.stork.network/api-reference/rest-api.

[172] “Holešky testnet shutdown announcement,” https://blog.ethereum.org/2025/09/01/holesky-shutdown-announcement.

## Appendix A. Oracle Capabilities in Practice (Extended)

We first discuss our methodology, followed by the results we obtained answering the questions and supporting the key findings discussed in Section 7.

## A.1. Methodology

**Selected blockchains**. We focus on general-purpose, permissionless smart contract-enabled blockchains, including naturally Ethereum. We also chose Cardano due to its unique design that fosters scalability and interoperability. Moreover, due to the growing rise of layer-2 solutions, we chose Polygon PoS [145], a popular layer-2 solution for Ethereum that many oracles support. This enables us to compare how an oracle behavior/overhead may vary between layer-1 (Ethereum) and layer-2 solutions (Polygon). Furthermore, we considered other popular blockchains: Algorand known for its scalable design, and Solana, an increasingly popular blockchain known for its high throughput and large market cap.

**Selected oracles.** The focus of our evaluations is on testing oracles that are deployed in practice, hence we do not cover academic oracles. Of the 19 industry oracles, we found that only 16 of these have usage information (i.e., deployed dApps that use them); TLSNotary is still early stage, and hence no usage information exists, whereas DOS and Provable were both found to be no longer active at time of analysis (while they were active during the stage of analyzing their design and documentation).

In terms of testing oracle capability, only 7 oracles, namely, Chainlink, Pyth, Tellor, UMA, Witnet, Stork and Supra, enable doing so. ZKPass and Gora are still early stage, and thus no testing was available on any of the selected blockchains. For Band, arbitrary URL queries, off-chain computation queries, and pre-defined data queries are available only on Band's app-chain with Cosmos, but not on any of the selected blockchains without custom bridging functionality. Thus, we could not test any of these functionalities for Band. Similarly, Redstone and Switchboard have no testing capabilities available on any of the selected blockchains' testnets.

For API3 and DIA, data feed availability is restricted to a permissioned set developed by their teams, and API3 further only implements a permissionless publish-subscribe pattern that requires a high subscription fee, thus we were unable to test any of these oracles. Finally, Charli3 and Orcfax only offer data from pre-defined immediate-read data feeds, but not request-based queries.

**Performance metrics.** For oracle use, we report the *number and types of dApps* that use an oracle, and the *number of query transactions* these dApps made for arbitrary queries/off-chain computation and immediate-read data feeds. Additionally, for the latter, we identify the number of active feeds offered by an oracle, and examine their *liveness/freshness*, i.e., whether data is actually updated at the promised frequency. For oracle capabilities, the quantification is related to verifying whether a *specific query type can be tested on the testnet*, and the *gas cost* of making such queries. We also measure the *response time*—the time it takes a dApp to receive a response for a request they made (averaged over 30 runs).[9]

**Experiment setup.** We use DefiLlama [12] to collect data on the dApps serviced by oracles. For on-chain transaction volume, we collect that from an oracle's OISC on each of the selected blockchains' mainnets using their explorers: Etherscan [146], Polygon Scan [147], and Cardanoscan [148]. For oracle capabilities, we conduct request-based experiments on the selected blockchains' testnets, i.e., Polygon Amoy [149] and Ethereum Sepolia [150] (for Cardano, the examined oracles did not offer request-oriented feeds). We implement our contracts (to submit queries and receive responses) in Solidity [151] using the Hardhat Development Framework [152] alongside Infura [153], an RPC provider service—allowing us to interact with the testnet, and a Metamask [154] wallet. For some experiments, we rely on external software offered by Chainlink, Supra, and Pyth to facilitate requests; namely, Chainlink's Functions Toolkit [155], Supra's REST APIs [156], and Pyth's Hermes [74].

TABLE 6. NUMBER OF DAPPS SECURED PER ORACLE.

| Oracle | No. of dApps secured | Percentage |
|---|---|---|
| Chainlink | 423 | 44.1% |
| Pyth | 270 | 28.1% |
| Redstone | 69 | 7.2% |
| API3 | 41 | 4.2% |
| DIA | 32 | 3% |
| Stork | 23 | 2.4% |
| Band | 22 | 2.3% |
| Switchboard | 18 | 1.9% |
| Supra | 14 | 1.4% |
| Witnet | 11 | 1.1% |
| Other | 36 | 3.75% |

## A.2. Limited Results for Solana and none for Algorand

Due to several reasons (which we identified while conducting our experiments), we were able to obtain only limited oracle usage data for Solana, and unable to obtain any data for Algorand

In terms of oracle usage on mainnets, only a single oracle, Redstone, was found to be active (so we able to examine its immediate-read feeds). First, for immediate-read data feeds, neither Gora for Algorand nor Chainlink/Supra for Solana have any active feeds according to their reference list of feeds [157]–[160]. Tellor while implemented on Algorand, reaching out to its development team, we found that it is no longer active/supported. Whereas for the two oracle's offering arbitrary query or off-chain computation support on these blockchains (Switchboard on Solana and Gora on Algorand), we were unable to obtain their OISC's addresses (and we did not hear back upon reaching out to their development teams).

In examining *oracle capabilities* on the testnets, though Gora appears live on the Algorand testnet, we could not find documentation/references for interacting with this version of the oracle. Instead, current documentation is limited to end-to-end testing, relying on self-proposing data on a local testnet that we have to deploy rather than the official Algorand testnet. Further, discussions with their development team did not help, they mentioned that they are still in the process of reconfiguring the oracle. For Solana, we were unable to perform our experiments due to a breaking change with Anchor [161],

9. While we additionally hoped to collect data on oracle fees from mainnets, the various OISC designs made it hard to collect such information. We also note fees on testnets are quite arbitrary rather than what an oracle actually charges on the mainnet, hence we do not report them.

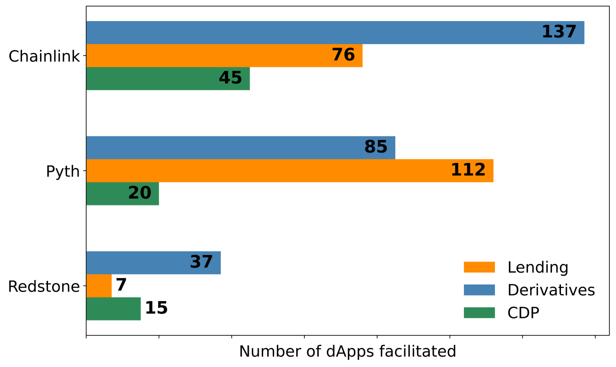

Figure 4. DApps secured by type.

Solana's primary development framework, in early 2025. While this issue itself was resolved [162], it persisted when interacting with external packages (e.g., SDKs for oracles). We will monitor this issue, and if resolved, we will extend our results accordingly.

## A.3. Results: Oracle Usage in Practice

**A.3.1. Application Support.** Application support is a strong indicator of oracle utility and practical impact. Using DefiLlama [12], we report the number of dApps secured by each oracle and the key industries represented. For oracles not supported by DefiLlama, we manually collect this data by inspecting their websites and prior partnership announcements.

As shown in Table 6, Chainlink, Pyth, and Redstone control a majority of the marketshare, around 79.4%. While we find 6 oracles each make up less than 1% (denoted as other in the table). These include Charli3 securing 9 services, UMA with 8, ZkPass, Gora, and Tellor each with 5, and Orcfax with 4. Inspecting the type of these dApps shows that almost all are within DeFi, and indeed for oracles with largest application support, we find the three leading sub-types for each include lending, derivatives, and collateralized debt position (CDP) services, as depicted in Figure 4. The prevalence these industries hold is natural given their reliance on external real-time data data for key functions (i.e., collateralization and liquidation procedures for lending and CDP systems, order executions for derivatives and options services). Oracles play a key role in offering such data usually via pre-defined data feeds.

Oracles with additional capability (beyond pre-defined feeds) cater to services in other industries; UMA, supporting human-based arbitrary queries, facilitates event-based prediction markets such as Polymarket [60], whereas ZkPass, that supports arbitrary queries with privacy guarantees, has been utilized in RegTech (regulatory technology) such as KYC verification [163].

**A.3.2. Oracle Transaction Volume.** Another metric of oracle usage in practice is the number of requests fulfilled. We examine that for immediate-read feeds, arbitrary queries, and off-chain computation requests (we find no separation between requests for arbitrary data and off-chain computation and thus report them together).[10]

TABLE 7. IMMEDIATE-READ DATA FEEDS PER ORACLE.

| Oracle | Ethereum | Polygon | Cardano | Solana |
|---|---|---|---|---|
| Chainlink | 142 | 164 | N/A | 0 |
| Redstone | 48 | 2 | N/A | 8 |
| Witnet | 0 | 4 | N/A | N/A |
| Orcfax | N/A | N/A | 20 | N/A |
| Charli3 | N/A | N/A | 7 | N/A |

**Immediate-read**. For these feeds, our investigation accounts for price update volume rather than end-use by dApps, i.e., when data is actually used on-chain versus being just updated. As immediate-read feeds can be read directly, they do not require transactions for that, making it challenging to examine their usage (conversations with Chainlink's developer team confirmed these challenges).

For all listed immediate-read data feeds for Redstone [158], Orcfax [164], Chainlink [159], Charli3 [165], and Witnet [166], we collected all transactions to each oracle's OISC during April 9-16, 2025 on Cardano, Ethereum and Polygon mainnets. Additionally, data for Solana was later collected during October 21-28, 2025—no such data was available during the initial time period. We found that 5 oracles had active feeds during this time, Chainlink and Redstone on Ethereum mainnet; Chainlink, Redstone, and Witnet on Polygon, Charli3 and Orcfax on Cardano; and Redstone on Solana. Table 7 shows our findings (for this and subsequent tables, 0 denotes no oracle activity while N/A denotes an oracle with no support on that chain). For both Ethereum and Polygon, Chainlink operates the majority of data feeds (around 74% and 98% of available feeds, respectively).

For each of these feeds, we report the volume of updates in Table 7. Against our belief that Ethereum would be the most popular, Polygon had a significantly higher number of updates, over 59x the gross volume than that of Ethereum—-almost all coming from Chainlink. We believe this is due to the cost advantages of Polygon, offering more inexpensive updates, and enabling numerous data feeds with heartbeats as low as 30 seconds. While for Chainlink, the average heartbeat is around 24 hours unless spurred by significant price deviation. For Cardano, despite that Orcfax offers around 3x the number of feeds compared to Charli3 (Table 7), Orcfax has around 35x the transaction volume of Charli3 (Table 8). We also found that Orcfax's feeds are being updated at a low heartbeat, on average every hour, compared to Charli3's 24 hour average heartbeat. Additionally, we find that Redstone's Acred_Solana/USD data feed on Solana reporting frequency does not match its specified heartbeat of 9 minutes and 50 seconds—reporting only 144 transactions (around 13%) of the expected 1061 given this heartbeat over a 7 day time period.

In terms of feed update liveness, we examined over 387 feeds and all were found active except Witnet; it was inactive during April 7-14, 2025. Witnet, however, retained activity so we examined it for additional two weeks during which all of its feeds were live.

**Arbitrary queries and off-chain computations.** We collect fulfilled request data by monitoring transactions for

10. We forgo examining transaction volume of request-response from predefined feeds given the growing prevalence of pull-based oracles, that employ off-chain requests, whose design complicates data collection.

TABLE 8. Immediate-read data feed transaction volume.

| **Oracle** | Ethereum | | Polygon | | Cardano | | Solana | |
|---|---|---|---|---|---|---|---|---|
| | # | % | # | % | # | % | # | % |
| Chainlink | 11779 | 69.3 | 695729 | 99.99% | N/A | N/A | N/A | N/A |
| Redstone | 5204 | 30.7 | 21 | <0.01% | N/A | N/A | 457 | 100% |
| Witnet | 0 | 0 | 286 | <0.01% | N/A | N/A | N/A | N/A |
| Orcfax | N/A | N/A | N/A | N/A | 4275 | 97.4% | N/A | N/A |
| Charli3 | N/A | N/A | N/A | N/A | 117 | 2.6% | N/A | N/A |

TABLE 9. Arbitrary query and off-chain computation transaction volume per oracle.

| **Oracles** | Ethereum | Polygon |
|---|---|---|
| Tellor | 4373 | 125 |
| UMA | 108 | 1991 |
| Chainlink | 90 | 49 |
| Witnet | 7 | 1465 |

each oracle's OISC during the period January 1 - April 30, 2025 over Ethereum and Polygon mainnets. Among the selected oracles, only $4$ support arbitrary queries and off-chain computations; Tellor, UMA, Chainlink, and Witnet. Our results are shown in Table 9. As shown, Tellor made up a majority of the volume on Ethereum mainet—$2204$ transactions, followed by UMA with only $160$ transactions. Whereas for Polygon, Witnet made up the majority of $3407$ transactions, trailed by UMA with $995$ transactions.

Albeit, the shown popularity of Tellor and UMA is primarily due to reliance on human-formulated arbitrary queries, where all transactions are viewed as arbitrary query or off-chain computation. This is not the case for other oracles for which query types can be cleanly separated. Nonetheless, the low number of arbitrary queries and off-chain computation requests points towards little production use for such query types; pre-defined feeds, instead, are dominant in terms of on-chain oracle usage.

## A.4. Results: Oracle Capability Evaluation

We conduct experiments to evaluate oracle capabilities on the selected blockchain's testnets. This enables us to examine not only maturity of oracle testing in practice, but also how oracles compare to each other in terms of gas overhead and latency, and even within the same oracle across different blockchains. Our experiments cover requests-response for arbitrary data and off-chain computations, and request-response from predefined-data feeds. Also, we examine privacy support for arbitrary data queries. We do not evaluate publish-subscribe queries as they are an extended variant of a request-response feed. For simplicity, we refer to Ethereum and Polygon's testnet by their names as Sepolia and Amoy, respectively. For all evaluations in this section, we examine only queries that can be made in a permissionless way, i.e., do not need pre-approvals from oracle providers.

*Testing oracle capabilities on testnets is lagging behind the actual support on mainnets.* While conducting our experiments, we observed that oracles' capabilities differ between a testnet and a mainnet of a particular blockchain. In other words, there is no full test support for all oracle request-based capabilities. For example, UMA does not support numerical responses on Polygon's tesntet, but it does that on the mainnet. Also, Tellor while not fulfilling our requests on Amoy, we find active usage on Polygon mainnet as discussed earlier. This highlights a critical gap; testing on testnets is a crucial component of secure and reliable dApp development and deployment. Without oracle testing capabilities, developers cannot fully test their dApps that use oracles, leaving these dApps vulnerable to attacks/security concerns that could be discovered after mainnet deployment.

**A.4.1. Arbitrary URL Queries.** We deployed a set of custom smart contracts, and chose a representative arbitrary URL query; current weather data. We used the open-source weather data source, Open-Meteo [167] to request this data (updated every 15 minutes) for Chicago, USA.

Of our set of selected oracles, four oracles support arbitrary queries in some form—Chainlink, UMA, Tellor, and Witnet. Witnet, while supporting permissionless arbitrary queries, requires the use of Witnet's SDK for query formulation; this SDK relies internally on the now deprecated [168] truffle development suite. Upon reaching out to Witnet's development team, they mentioned an ongoing work on upgrading the SDK to remove this dependency. Thus, we could not test Witnet. Tellor, on the other hand, does not offer permissioned arbitrary queries as testing them requires manual integration by the Tellor team. This leaves us with two oracles, UMA and Chainlink, to examine.

On Amoy, as UMA supports only arbitrary queries with T/F answers, we adapted our query to be whether the current temperature in Chicago USA was above $10$ degrees celcius. In testing UMA on Amoy and Sepolia, we chose a $15$ minute request duration; $10$ minutes to wait for a response followed by $5$ minute challenge period. We self-propose response data, if not proposed by a data feeder after waiting a $10$ minute liveness period. Due to its optimistic approach, UMA allows self-fulfillment without losing guarantees of accuracy, given a dispute party is still incentivized to verify this data. In terms of gas consumption, we report that for both with and without self-proposing. We note that none of our query tests received a data feeder response, and thus we ended up self-proposing all responses (we waited for 10 minutes before doing so, and thus no data feeder response latency is available). In terms of gas cost, an arbitrary request for UMA on Sepolia consumes $451,710$ (it would be $303,387$ units without self-proposing since we have to pay the transaction fee for posting the response). On the other hand, we find the T/F-adapted query on Amoy consumes $413,983$ (costing $288,562$ units without self-proposing).

Whereas for Chainlink, a request resulted in $279,075$ units on Sepolia taking 21 seconds, and $275,824$ on Amoy taking 10 seconds. To do the testing, we note that we needed LINK test tokens (while UMA did not need any test tokens). In addition, upon deploying our test contracts, we had to interact with Chainlink's frontend [169] to create a new subscription to pay for our subsequent data requests. These steps needed only for deploying a new contract, rather than for each request. As noted, overall Chainlink testing capabilities are more active, and it offers cheaper responses, than UMA.

**A.4.2. Privacy Support.** We examine oracle privacy support for arbitrary queries. As discussed in Section 5.4, privacy for oracles today mainly revolves around I/O

TABLE 10. INPUT PRIVACY SUPPORT OVERHEAD IN CHAINLINK.

| **Approach** | Ethereum (Sepolia) | | Polygon (Amoy) | |
|---|---|---|---|---|
| | Gas units | Time | Gas units | Time |
| DON-hosted | 295,024 | 26.1s | 291,773 | 10.5s |
| Self-hosted | 298,723 | 26.3s | 295,468 | 11.26s |

privacy. On output privacy, we note that it is not yet viable in practice (only Deco and ZKPass offer that; the former is not deployed yet while the latter is still in early stages not suitable for testing). Thus, we focus on input privacy.

We find that only Chainlink supports this notion, and it does so using two different means: (1) encrypting secret inputs under the DON's public key, or (2) self-hosting these secret inputs (e.g., in a Github Gist [170]) while encrypting a reference to this location under the DON's public key to be sent as part of the request. We test both approaches where we treat the Open-Mateo URL as a secret URL that we want to keep hidden.

Our results are shown in Table 10. Comparing Gist to DON-hosted secrets, we find that the latter tends to be fulfilled on average slightly faster. Whereas comparing our findings to arbitrary requests without input privacy, we find requests with privacy tend to take slightly longer (approximately 1.2x the public variant on Sepolia), and consumes approximately $18,000$ more gas units on average. This is natural as privacy support requires additional parameters and are more complex. In particular, additional flags must be passed into the request indicating that some parameters are private as well as whether the private data is self-hosted or sent directly to the DON, in addition to the encrypted query. Thus, the smart contract logic dealing with these private queries is more complex and requires additional gas cost. Similarly, as there is now an extra step of retrieving the secret input (either decrypting directly or self-hosted), this leads to additional delays.

**A.4.3. Off-chain Computation Queries.** Building on our use case of weather data, we create a computational script to calculate the historic monthly temperature for two U.S. cities, Seattle Washington and Portland Oregon, again based on Open-Mateo data source (from historic hourly temperature data for the month of January 2023), and then return the city with the lowest temperature.

Three oracles support off-chain computation queries: Chainlink, UMA, and Tellor. However, Tellor require pre-approval from its team, as mentioned above, so we test Chainlink and UMA. For UMA, we translated the problem to a question providing the data sources and computation instructions, whereas for Chainlink, we wrote the script in JavaScript as required. In making a data request, for UMA this question is provided on-chain within the request, and for Chainlink, an encoding of the script is sent on-chain to be relayed to the DON.

For UMA, we once again chose a $10$ minute liveness period followed by a $5$ minute challenge period. As before, none of the feeders responded, so we had to self-propose the response after the liveness period is over. On Sepolia, the average request gas cost is $453,936$ units (which would be $303,387$ units if not self-proposed). On Amoy, the results were similar; $464,117$ units for self-proposed responses (which would be $322,668$ units if not self-proposed). In comparison, Chainlink on Sepolia consumed $365,250$ units taking $21$ seconds to fulfill a request, whereas on Amoy, it consumed $361,999$ units taking 8 seconds. As noted, UMA (without self-proposing) and Chainlink have quite similar costs. In terms of latency, self-proposing aside, UMA's need for a challenge window leads to a slower fulfillment time. Whereas comparing Chainlink's performance among chains, we once again find requests on Amoy are fulfilled significantly faster than those on Sepolia.

TABLE 11. PRE-DEFINED FEEDS REQUEST-RESPONSE SUPPORT.

| **Oracle** | Ethereum (Sepolia) | | Polygon (Amoy) | |
|---|---|---|---|---|
| | Gas units | Time | Gas units | Time |
| Pyth | 246,229 | 0s | 247,092 | 0s |
| Supra | 388,325 | 0s | 388,434 | 0s |
| Tellor | 386,762 | 10:03mins | Not fulfilled | N/A |
| Stork | N/A | N/A | 93,861 | 0s |

**A.4.4. Pre-defined Data Feeds.** Finally, we conduct an experiment examining request-response style queries from pre-defined feeds. Given the primary use of pre-defined feeds in reporting financial data, we examine oracle ability to retrieve price data for the BTC/ETH price pair. If an oracle does not support BTC/ETH, but supports ETH/BTC we use this pair price instead. If neither exists, we request the ETH/USD and BTC/USD price pairs and calculate BTC/ETH price based on that. To better highlight usage in practice, i.e., the reliance on real-time data, we set a 20 minute fulfillment window for the requests. As the underlying data source varies from oracle to oracle, we forgo an analysis of price discrepancy (for refer the reader to prior work on this issue [48], [50]).

Among the selected oracles, 6 of them support request-response queries from pre-defined datasets; Chainlink, Stork, Charli3, Supra, Redstone, Tellor, and Pyth. However, testing was viable for only 4 of them; Stork, Supra, Tellor, and Pyth. Charli3, while recently has announced pull-based oracles from pre-defined data feeds [71], is still in closed early stage. Similarly, Chainlink's pull-based oracles are still in early-stage and require an authentication token to fetch off-chain data; we did not hear back from Chainlink's team upon reaching out to obtain this authentication token. Lastly, Redstone does not currently support testing on Amoy or Sepolia.

Pyth, Supra and Stork required off-chain data requests, i.e., first collect price data off-chain using their REST APIs [74], [156], [171], and then send this data on-chain alongside the request to be used on-chain after a quick verification that the data came from their API (leading to almost zero response delay). For all four oracles, the BTC/USD and ETH/USD pairs were used (recall we are mainly interested in BTC/ETH price). Batching on Pyth, Supra, and Stork allowed us to update the price within a single transaction, whereas Tellor required two transactions. Tellor additionally required the use of TRB tokens to facilitate the requests; for Amoy, Tellor operates a faucet to obtain TRB tokens, while on Sepolia, we had to reach out to its development team to obtain test tokens.

Table 11 shows our findings. Pyth and Supra both successfully retrieved update price data on both Amoy and Sepolia, whereas Tellor only did that on Sepolia—although data requests were sent to the data feeders on

Amoy, we did not receive any responses back.[11] On the other hand, Stork has no support for Sepolia; it used to have support on another Ethereum testnet, namely Holesky, which is now deprecated [172]. In comparison to the results in Table $11$, we find that Stork offers the best performance at roughly $0.39$ the cost of the next lowest Pyth, and around $0.24$ the cost of both Supra and Tellor. Moreover, of the $30$ runs performed we note that Tellor fulfilled the request for only $15$ runs.

To contextualize these results, we additionally examine how they compare to a similar request made from oracle supporting arbitrary queries, i.e., Chainlink. Utilizing the Crypto-Compare API to retrieve the BTC/ETH price, we reran our experiment with Chainlink retrieving the same price. Our findings align with those in Section A.4.1; this price request consumed a total of $274,854$ gas units on Amoy and $278,104$ units on Sepolia. Making it less expensive than Supra and Tellor, but still roughly $2.92x$ the cost of Stork, and $1.12$x the cost of Pyth.

11. Reaching out to Tellor's team, we were told that data feeders on the testnets need reconfiguration in order to fulfill requests. After doing so, we were able to retrieve data on Sepolia, but not on Amoy.