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*—Oracle services enable dApps to access real-world data and respond to events happening outside the blockchain ecosystem. A large number of academic and industrial oracles have emerged, capturing various designs, capabilities, and security assumptions/guarantees. This rapid development makes it challenging to comprehend existing solutions, understand their trade-offs, and build on them.**

In this paper, we introduce a systematization of knowledge for blockchain oracles. We develop a multi-dimensional systematization framework assessing existing solutions based on their capabilities, trust and security assumptions/guarantees, and their design architectures. To further aid in this assessment, we go on to distill a number of insights and gaps, and provide a guide for practitioners (on the use of these oracles) and researchers (by highlighting future research avenues).

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 chains [10], [11], timely interaction with the real world 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, they 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]–[40]. These vary based on their capabilities (in terms of supported queries), security and privacy guarantees, trust assumptions, and implementations. Also, oracle designs range from theorized ones limited to academic works to those deployed and widely used in practice.

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 oracle designs and implementations impact these capabilities?
- **RQ2.** What are the security and privacy guarantees of existing oracles?
- **RQ3.** What are the existing gaps, future research directions, and recommendations for oracles' use in practice?

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 with an empirical investigation of their capabilities.The utility of our work is three fold: simplifying the task for newcomers to understand the landscape; offering a guideline for practitioners when choosing oracles for practical applications; and paving the way for new research avenues 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, namely, 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. We also identify an interconnected relationship between these capabilities and oracle-dApp interaction patterns. Additionally, among industrial oracles deployed in practice that offer multi-blockchain support, we find that most of them rely on separate deployments on each blockchain, while a small set avoid that by employing inter-blockchain messaging. In terms of response authenticity, the majority of oracles utilize statistical aggregation or optimistic approaches. We also identify a trend of limited privacy support primarily pointed towards input privacy.

Distilling insights and identifying gaps. Throughout our analysis, we distill numerous insights and call attention to a number of gaps with respect to current deployments as well as open research questions. We also offer a guide for practitioners on how to choose oracles for their applications. 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 (and other Ethereum virtual machine (EVM)-based chains).We also discuss oracle implications in terms of connecting blockchains to legacy systems, relation to maximal extractable value (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. Existing SoKs and surveys around oracles either study a small number of oracles, or focus on these tied to a single blockchain or particular usecases. Moreover, many of them are outdated given the rapidly changing landscape.

Beniiche [41] studies oracles in terms of their interaction patterns, (de)centralization aspects, and data source types, covering only Provable and Chainlink. Al-breiki et al. [42] focus on oracle components of trust, Zhao et al. [43] study the design and trustworthiness of oracles used in DeFi (namely, Chainlink, Band, NEST Protocol [44], DOS Network, and Witnet), whereas Ezzat et al. [45] view oracles from an interoperability perspective examining 8 oracles alongside other messaging protocols [46] and multi-chain platforms [47]. Pasdar et al. [48], [49] examine a number of general-purpose and usecase-specific oracles and blockchain platforms. However, they focus on how oracles establish confidence in responses, including voting-based and reputation-based approaches, rather than a comprehensive synthesis of oracle designs, capabilities, and security. Eskandari et al. [50] cover 17 oracles, studying their workflow and operation while highlighting security risks. 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 [51], as oracles.

Not only does our work provide an up-to-date view of the rapidly evolving oracle landscape, it also stands out in two key ways 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 oracles spanning various blockchain architectures, (3) Distills a number of insights and current gaps in the current oracle landscape, moreover expounding on this, we map a number of future research directions according to these identified gaps.

2. Background

To facilitate the discussion, we review the general design architecture 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.

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

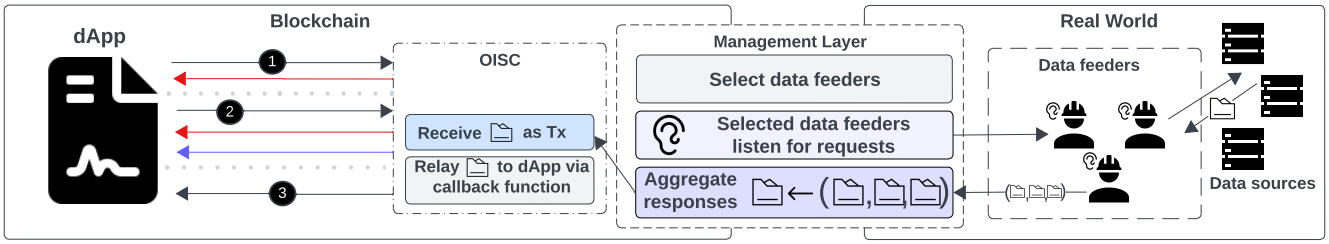


Figure 1: Oracle design architecture (interaction patterns indicated by ball numbers: ❶ = request-response, ❷ = publish-subscribe, and ❸ = immediate-read).

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. To accomplish this, off-chain data feeders are tasked with querying and returning data in accordance with the update criteria specified in the subscription contract. 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, depending 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 general-purpose oracles including peer-reviewed academic works and popular industrial solutions. We do not attempt to describe in detail how these oracles work, but rather contextualize 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" while examining the first 20 search result pages from Google Scholar, IEEE Explore, ACM Digital Library, and USENIX. Works that: (a) develop usecase-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 other existing designs, are out of scope. Accordingly, our refined search identified 5 academic oracles, including Town Crier [22], one of the first works on authenticated oracles via trusted hardware; DiOr-SGX [28] which extends Town Crier with a distributed version; DAON [52] that adds a reputation mechanism; Astraea [21] that proposes a stake-based approach to decentralization; and DECO [23], a decentralized and privacy-preserving oracle.

For *industrial works*, we include oracles operating on various blockchains, both EVM and non-EVM. We focus on active oracle solutions with public-facing documents; whitepapers, documentations, and source code. 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 highest 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]. As of April 2026, Chainlink, Pyth, and RedStone account for 75% of total oracle TVL, around 50.1 billion USD [12], while API3, ranked last on our list, has a TVL of around 44 million USD. For historically popular oracles, i.e., those were popular in the past but their current TVL is not large, we include additional 4 oracles; Provable [16], Tellor [18], DOS Network [34] and Witnet [36]. Interestingly, among these 14 oracles, none is being operated on Cardano or Algorand. Thus, we include relevant oracles for both; Orcfax [53] and Charli3 [54] for Cardano, and Gora [55] for Algorand. To round out our examination, we finally select industrial works that feature unique oracle designs, including TLSNotary [31] and ZkPass [26] for their response-authenticity techniques.

Figure 2 traces the development timeline of the selected oracles, and for industrial ones, it includes their launch on popular blockchains.[1]

Systematization framework. We develop a systematization framework that distills design characteristics and distinguishing features of the studied oracles. It covers 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 pre-defined 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. The latter includes cryptographic assumptions, trusted hardware, trusted third parties, and economic security.

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

Privacy 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 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 feature include employing inter-blockchain messaging protocols and deploying separate oracle instances. Lastly, we discuss the various fee and rewards structures used by

1. For academic works, the works [56]–[58] target the game theoretic aspects of Astraea's model; due to their similarity in design to Astraea, we forgo discussing them. For industrial oracles, we forgo Chronicle [25] since it only offers price feeds to a permissioned set of dApps, and Edge [59] due to limited documentation. Also, Truebit [15] is a prior oracle for off-chain computations—this older version is inactive (a new rendition is being developed [60]). Moreover, in the final stages of this work, we found that Provable and DOS were no longer active (no official announcements of deprecation).

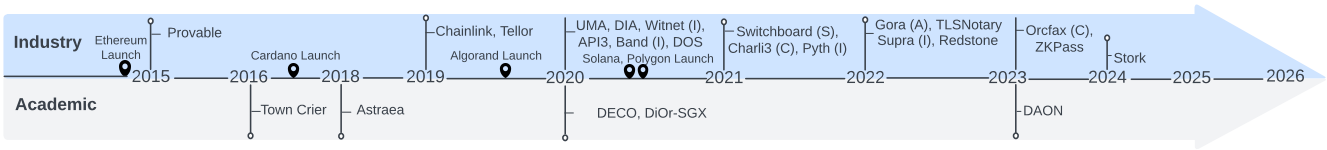


Figure 2: Evolution timeline of the selected oracles (we specify launch on Ethereum without parentheses after an oracle name, while we use (A), (C), (S), (I) for Algorand, Cardano, Solana, and app-chains, respectively).

oracles, i.e., how they reward data feeders and what/how dApps pay for oracle queries.

4. Oracle Capabilities

Being 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 these three dimensions (see 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. 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), GDP, personal consumption expenditures (PCE) price index, and 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.

The specific data available varies from oracle to oracle. As expected, widely used data types are usually offered by all oracles, e.g., BTC/USD price. Also, for oracles with multi-blockchain support (see Table 5), their pre-defined data sets may vary across different blockchains. Chainlink, Redstone, and Witnet have 183, 70, and 1 immediate-read data feeds available on Ethereum, whereas 142, 4, and 14 feeds on Polygon, and 212, 22, and 2 feeds on Arbitrum, respectively. Availability decisions are highly impacted by the mechanisms used for multi-blockchain support, which we discuss in Section 6.2.

> ***Insight 1.*** 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.

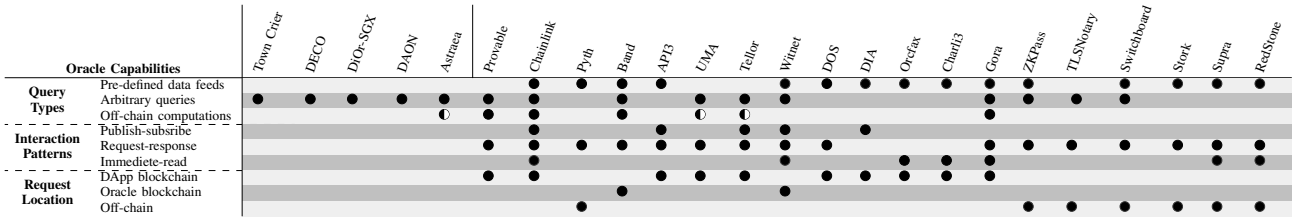
Arbitrary queries. This class enables a dApp to request data from an arbitrary source; either an arbitrary URL (web-based) or any party that can answer a question of interest (human-based). It is a flexible class usually used for unpopular/new feeds, e.g., prices of emerging digital assets or results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

> ***Insight 2.*** Arbitrary queries address trust concerns in pre-defined feeds by allowing dApps that don't trust a pre-defined data source to specify custom sources. However, in doing so, they also create new attack surfaces: ambiguous query parsing, malicious or unreliable source selection and the lack of systematic data feed validation present in curated data feeds.

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 [61], 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

TABLE 1: Oracle classification based on their capabilities (◐ = conditional support).



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 (e.g., UMA-related disputes within Polymarket [62], [63]).

Off-chain computations. Under this class, oracles facilitate dApp-specified computations off-chain. As shown in Table 1, we find that 7 oracles support off-chain computations—Provable, Chainlink, Band, Gora, Astraea, Tellor, and UMA.

A computation query is composed of a script and its inputs. Oracles specify scripting languages that can be used (C++ for Provable, Javascript for Chainlink, Rust for Band, and C for GORA where for both Band and Gora, the script is compiled into WASM), as well as limits on script configuration and execution. The latter includes maximum allowed values for runtime, script size, input/output size, and the number of HTTP calls. Often the script is either posted on external storage sites such as IPFS [64] as in Provable or the oracle's app-chain as in Band, or sent directly to the data feeder network (decentralized oracle network—DON) as in Gora. Whereas for Chainlink, scripts are sent as part of the data requests on-chain (like other requests that will be routed to the DON). The inputs also can be passed-in parameters in the query or retrieved via an external call to a data source. In the case of Chainlink, to support input privacy, secret input parameters can be sent directly to the DON. Upon receiving the script and inputs, data feeders in Chainlink, Band and Gora execute the script individually, agree on the final output via a majority-based consensus, and then the output is returned to the OISC. Band and Gora allow the computation request issuer to specify an aggregation method (other than the majority) at the time of request. Provable, being a centralized oracle, runs the script once and returns the output.

In principle, human-based oracles (UMA, Tellor, and Astraea) can support off-chain computations; the question contains computation steps and the inputs (Table 1 denotes this conditional capability with half-shaded circles). Computation questions must be carefully formulated as the human-interpretable nature may lead to new ambiguities not found in scripts. Also, they are limited to publicly-verifiable deterministic computations (to enable disputing the output). Furthermore, output type could be restricted; Astraea supports T/F questions, and so only computations with binary output. Whereas UMA and Tellor support both T/F and numerical responses, thus offering a wider range of computations.

> ***Insight 3.*** We believe that the increased adoption of layer 2 scalability solutions, e.g., rollups [65]–[68], and the recent emergence of dedicated co-processors that enable performing expensive computations off-chain [69], have shifted the focus of oracles away from supporting computation queries. This is supported by, where 5 of the 6 industrial oracles supporting arbitrary computation were launched before 2021, with the majority of recent works focusing instead of low-latency oracles from pre-defined data feeds.

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.

Although each query type can be facilitated using any of the patterns discussed in Section 2, we find that oracles choose specific patterns based on data feed type. When an oracle offers multiple interaction patterns, selection is influenced by the underlying dApp use case, e.g., response frequency or that certain data types are available via certain patterns. In general, 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 immediate-read. Moreover, Chainlink, Witnet, Tellor, API3, and DIA employ publish-subscribe 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; Chainlink, for instance does not offer immediate-read on all the supported chains, and only the pull-based request-response pattern is available on Solana, Sei [70], and Unichain [71].

Insight 4. Interaction patterns are tied directly to the specific data type being offered. Namely, immediate-read patterns are only offered for predefined 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. We find that 10 oracles follow this approach (Table 1). Under multi-blockchain support, 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 oracles 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 [72] 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 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 5. App-chains promote multi-blockchain support but could increase overhead; messaging protocols require additional transactions (to relay requests/responses) which in turn may lead to additional response delay (or latency).

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 employ this paradigm (Table 1). TLSNotary, ZkPass and DECO develop a third-party verifiable TLS variant to authenticate data feeds. While Pyth,

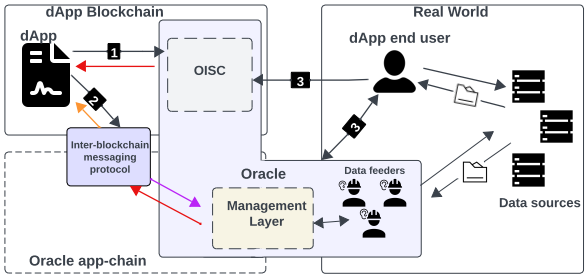


Figure 3: Oracle model by request location; dApp blockchain **1**, oracle app-chain **2**, and off-chain **3**.

RedStone, Stork, Supra, and Switchboard utilize a pull-based approach where feeds are posted/updated on an off-chain module. Requests directly pull responses and authenticity attestations from this module and submit them on-chain for verification and dApp usage. Table 2 summarizes how various oracles implement the pull-based approach. We note that Charli3 and Chainlink have announced a future upgrade offering pull-based oracles [73], [74] and as of 2026 Pyth has migrated its pull-based architecture [75] (we summarize both in the table).

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 aspects 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 examined oracles based on these dimensions (Table 3); this is after discussing existing oracle security notions.

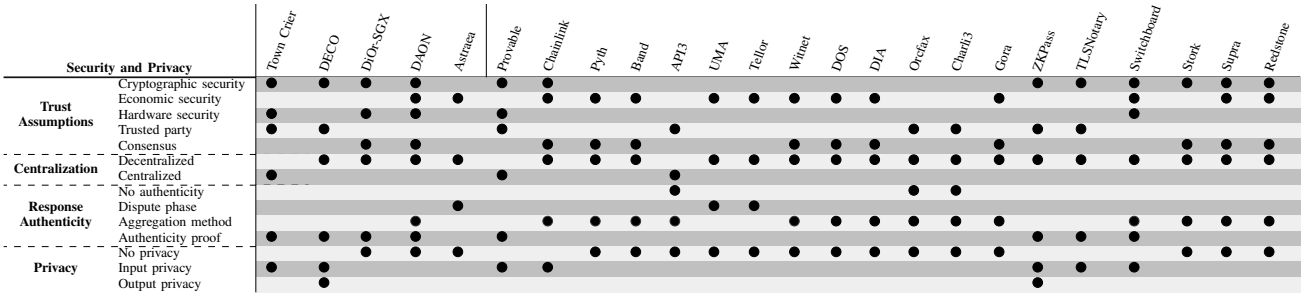
5.1. Oracle Security Notions in the Literature

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

TABLE 2: Pull-based oracle's data delivery architectures. →= migration in architecture

Oracle	Data Layer	Attestation Mechanism	Retrieval Method	Verification	Aggregation
Pyth	App-chain (Pythnet) → None (continuous WebSocket stream)	Wormhole guardians attest to Pythnet data → trusted independent-operators sign individually; requiring quorum	Hermes [76] (a web service) → Hermes and Pyth Pro (WebSocket Stream)	Cross-chain message verification (Wormhole → set of independent operators)	Pre-aggregated on Pythnet → Aggregated per operator then requiring 3 of 5 quorum among operators
Supra	App-chain (Supra L1)	Supra validators attest to data on Supra	gRPC server	On-chain verification contracts	Pre-aggregated on Supra
RedStone	Arweave-based data layer [77]	Individual data feeder signed data	REST API (DDL gateway)	On-chain verification contracts	None; Individual signed data points
Stork	None (continuous WebSocket stream)	Publishers sign individually, then aggregators sign aggregated values	WebSocket stream	Per-chain verification contracts	Done by aggregator parties
Switchboard	None (on-demand; no persistent layer)	TEE attestation + data feeder signed attestation	REST API (gateway to feeders' TEEs)	Per-chain verification contracts	None; feeders' TEEs individual responses

TABLE 3: Oracle classification across security and privacy dimensions.



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.

As noted, both notions are tied to the TLS oracle specific design they introduce. General purpose oracles and oracles deployed in practice have additional modules, thus their security requires different modeling and additional properties to be achieved. That is, an oracle app-chain, like a regular blockchain, must satisfy liveness and safety [78], [79]. Those that employ pull-based approaches must ensure the integrity of the off-chain modules that record responses and attestations. Oracles with economic security, where feeders that report incorrect responses are punished financially, require properties ensuring that punishments indeed deter misbehavior. We elaborate on these points further in Section 8.

Gap 1. Current security definitions of oracles are limited to only a subset of current designs. Defining a generic and formal security notion for oracles is still 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 in oracle designs. Digital signatures are inherent to preserve integrity, and used for response authenticity. That is, responses are signed by the TEE as in Town Crier, or by the feeders in the DON as in Chainlink, DIA, DOS, DAON, Supra, RedStone, Stork and Gora. Town Crier and DiOr-SGX utilize the default TLS protocol, while DECO, TLSNotary, Provable and ZKPass construct a third-party verifiable variant that employs a two-party computation (2PC) protocol. Also, Provable and TLSNotary support TLS

1.0/1.1 and TLS 1.2, respectively, while the rest support TLS 1.2/1.3. Lastly, DECO and ZKPass employ zero knowledge proofs (ZKPs) to prove statements over private data.

Trusted hardware. TEEs, or trusted hardware, are used to perform critical operations, mainly enforcing the correct behavior of 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, with the TEE-signed response 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 comes with strong assumptions on resistance to reverse engineering and side-channel attacks; several works [80]–[85] have demonstrated that TEEs are vulnerable to such attacks. Compromising a TEE enables the host feeder to forge responses. In fact, reducing the impact of such attacks is the main motivation for employing several TEEs (usually chosen from different manufacturers).

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) while avoiding middlemen feeders who may misbehave.

> ***Insight 6.*** 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 offer data on-chain.

Economic security. Economic security relies on rational incentives to deter misbehavior. This is done by making the attack cost, or the punishment upon detection, exceed the gains of the attack. In other words, economic security is achieved if the minimum cost to break a security property is larger than the potential gains resulting from the attack.

Many oracles rely on economic security to incentivize honest behavior of data feeders. Upon joining the system, a feeder puts in a collateral, that will be slashed for incorrect responses or lack of activity (with an eventual removal from the system if a feeder continues to misbehave). As shown in Table 3, 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 [86] 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 in their blockchains. The digital asset of the collateral varies between oracles, as shown in Table 4, and generally it is an oracle-operated token.

TABLE 4: Classifying oracles that employ staking-based economic security aspects (Pct = percentage, N/S = not specified, and SB is Switchboard). UMA has an oracle-operated ERC-20 token, called UMA, used for governance, which can be used for the collateral.

	Astraea	*DAON*	*Chainlink*	*Pyth*	*Band*	*Tellor*	*Witnet*	*DOS*	*DIA*	*Gora*	*SB*	*Supra*	*Redstone*	*UMA*
Collateral asset	N/S	N/S	LINK	PYTH	BAND	TRB	WIT	DOS	DIA	GORA	SWTCH/ JitoSoL	SUPRA	ETH/ RED	Any ERC-20
Penalty type	Variable	Other	Fixed	Pct	Pct	Fixed	Variable	Fixed	N/S	Other	Other	N/S	Other	Fixed

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, e.g. [62]. Moreover, reporting volatile data (such as digital asset prices) may lead to accuracy issues—value might change between the time of retrieval and that of delivery.

> ***Insight 7.*** 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.

Another important factor is the financial penalty bound required to deter malicious actors. We find that this also varies across oracles (see Table 4); 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. UMA and Witnet determine penalties on a feed-by-feed basis, which is left to the data feed creator to decide.

> ***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 [87], [88]. 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 8.*** 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 3) 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.[2] 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 3. 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 [90]) and Android hardware attestation tooling. Ledger proofs rely on a hardware wallet called Ledger [91] to attest to code execution by its TEE.

> ***Insight 9.*** 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 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 3), 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.

2. Supra utilizes a technique from [89] 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.

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 requires a party to pass (personal) authentication material when calling a data source API (e.g., to check that the party has a paid subscription to request data). Thus, the user/dApp must provide this material to the oracle so it can request the data. This sensitive material, and in some instances the entire request, should be hidden from the oracle to avoid deducing information about the dApp operation upfront.

Only 4 oracles, Chainlink, Provable, Switchboard, and Town Crier, support input privacy. Chainlink uses threshold cryptography where requests are encrypted under a DON's public key with the decryption key shared among the DON data feeders. A threshold k of the feeders are needed to decrypt the request then respond (Chainlink documentation does not provide in-depth details about this process). Thus, there is an implicit assumption that feeders are trusted not to disclose the request. Provable allows users to encrypt only one input (usually the authentication material) under the public key of its centralized data feeder (so Provable is trusted with this material). Finally, in Town Crier and Switchboard, requests are 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 [92], [93], 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 multi-blockchain support, and fees/rewards models. Table 5 classifies the examined oracles based on architecture and blockchain-agnosticism, whereas Table 6 classifies them based on fees and rewards. Studying these aspects is beneficial for future oracle design, or improving existing ones if possible, in terms of building 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).

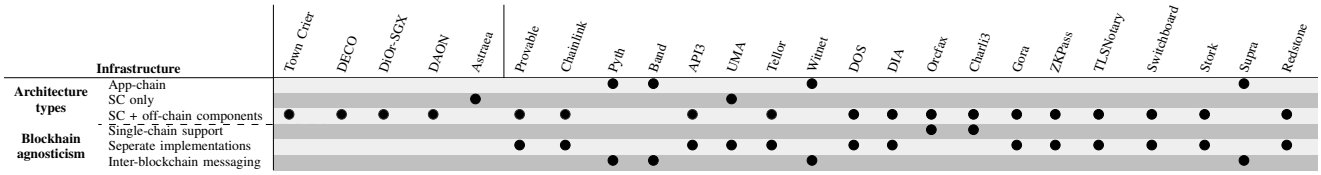
App-chain based. An app-chain is a blockchain centered around implementing the oracle functionality and its management layer. Interaction with this app-chain, to receive requests and relay responses, is done using inter-blockchain messaging protocols. We find 3 oracles (Witnet, Pyth, and Band) operate as PoS app-chains (Tellor recently announced Tellor Layer [94] a planned app-chain within Cosmos [95]). We note that as of 2026, Pyth is actively sunsetting its app-chain for a off-chain continuous data stream model as discussed in Table 2 The app-chain validators could be also tied to the oracle modules (rather than just running the consensus/network protocol). Both Witnet and Pyth (historically) 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 or 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 5). 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.

TABLE 5: Classifying oracles based on architecture and multi-chain support (SC = smart contract).



Insight 10. 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 of target specific chains; primarily Ethereum and other EVM-based chains. Academic oracles do not specify their particular blockchain support as their designs are posed in a generic way. As shown in Table 5, 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 as we discuss below.

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. Up until 2026, Pyth utilized Wormhole [96], recently adapting to protocol-specific messaging operated by permissioned independent operators, and Witnet uses an internally developed protocol, whereas Band employs IBC [72]. As a result, each oracle may support a varying blockchain set based on these protocols communication capability, i.e., the blockchains they support. In particular, Witnet, Band, and Pyth support 28, 21, and 90+ blockchains, respectively. Supra, which also operates as a general-purpose blockchain and its oracle has the same name, additionally utilizes an internally developed inter-blockchain messaging (supporting 58 blockchains) for oracle data first posted to its layer-1 and desired elsewhere.

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 5). 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 4 for a corresponding list of tokens of each oracle). Thus, when deploying several instances, these oracles would require asset bridges to manage equivalent token representations across chains. This adds design complexity, and may lead to requiring different incentivization mechanisms across separate oracle instances.

Insight 11. 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 [97]. While multi-instance deployment leads to silos, magnifying blockchain

TABLE 6: 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 (Var = Variable, UB = native asset of underlying blockchain, SB = Switchboard).

	Aeruna	DAON	Provable	Chainlink	Pyth	Band	API3	UMA	Tellor	Witnet	DOS	DIA	Orao	Chain3	Gora	SB	Supra	Redstone
Fee Asset	N/S	N/S	UB	LINK	UB	BAND	UB	Any ERC-20	TRB	UB	N/S	UB	FACT	N/S	GORA	SWTCH	SUPRA	UB
Fee Type	Var.	Var.	N/S	Var.	Fixed	Fixed	Var.	Fixed	Var.	Var.	Fixed	N/S	Var.	Fixed	Var.	Var.	N/S	Var.
Reward Asset	N/S	N/S	UB	LINK	PYTH	BAND	API3	Any ERC-20	TRB	WIT	DOS	DIA	FACT	C3	GORA	SWTCH	SUPRA	UB

interoperability problems.

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). 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 latter, 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 6). 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 oracles (5 oracles as shown in Table 6) adopt simplified fee models; usually fixed fees per request or over some period of time—in what might be considered as an approach to garner more customers.

Typically, fees are either paid in the native token of the blockchain where a dApp lives, or in the oracle-operated token (Table 6). 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, and generally these rewards come from the fees paid by dApps. Rewards could be in oracle-operated tokens (e.g., ERC-20 tokens), and in the case of app-chains digital assets underlying the oracle blockchain. We find the reward models fall under 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-chain-based oracles like Witnet, Band, and historically Pyth. The exact reward specifications may change over time and could depend on specific system tokenonomic features, such as inflationary/fixed asset reserve and disbursement policies, and 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. Choosing Oracles for Applications

Given the complex landscape of oracles, selecting an oracle for a particular application can be challenging. To help in this process, we synthesize the main decision points from a dApp developer's perspective, and offer guidelines based on the prior sections.

Q1: Where will the dApp be deployed? This is a critical question that impacts subsequent decisions. As most oracles target Ethereum, deploying on other blockchains, as of now, may limit developer choices and even the viability of the dApp itself. Lack of oracle support may drive the developer to target a different blockchain, or consider interoperability solutions to enable interacting with oracles active on other chains.

Q2: What external data is required? Common data, such as prices of major assets, are generally available by most oracles using pre-defined feeds. While exotic market data or non-financial data require oracles with arbitrary query support. When off-chain computation over data is required, the oracle space is even narrower as discussed in

Section 4. Not to mention that particular data, such as those from legacy systems, may require establishing partnerships and deploying pull-based feeds to obtain them.

Q3: What are the timing requirements for responses? For applications requiring high-frequency real-time data, pull-based oracles are the canonical choice, providing on-demand data fetching in one transaction. For data needed at regular intervals and some bounded stalelness is acceptable, immediate-read feeds are a suitable option. However, they are only offered for pre-defined feeds and their update frequency may vary across blockchains. In our empirical evaluation, we find a relationship between lower transaction costs and increased heartbeat (e.g., those on Polygon were updated more often compared to those on Ethereum). In turn, for applications operating on different blockchains, such factors may lead to, e.g., cross-chain arbitrage opportunities across dApp instances. When a dApp needs periodic data for a known time horizon, and without establishing off-chain agreements with oracles, publish-subscribe patterns are a desirable option.

Arbitrary URL and computation queries offer less flexibility; both appear exclusively in the request-response pattern with less control over timing. For example, oracles operating as app-chains require additional transactions for cross-chain messaging. While optimistic oracles require long challenge periods and possibly dispute resolutions (e.g., 2 hours for UMA in Polymarket [98]), lengthening their time to finality. Hence, they are more suitable for one-shot high-valued queries (e.g., insurance payouts or prediction markets payouts).

Q4: What oracle trust assumptions are acceptable? Oracles adopt various trust models, which we discuss in a sequence ordered by trust assumption strength.

Data source itself is the reporter. First-party oracles operate as a combined trusted data source-feeder. This tightly-managed design is appropriate when the dApp inherently trusts the data source, e.g., a partnered data provider.

Permissioned-set of data feeders. Several factors impact this choice, such as the number of data sources being contacted, the number of data feeders, response aggregation method, etc., which vary across oracles. For example, Chainlink's Ethereum ETH/USD price feed has 31 feeders, while its SOL/USD feed has 16, and both utilizing median-based aggregation [99]. (Chainlink does not publicly disclose the exact data sources nor their number [100].) In contrast, Band has 40 data feeders (validators on Band Chain) reporting data for both its ETH/USD and SOL/USD feeds, with ETH/USD being reported from 7 data sources and 9 for SOL/USD [101]. Both of these feeds utilize stake-weighted median aggregation.

Economic security. Optimistic decentralized approaches that employ economic incentives is another option to consider. They improve availability where any party can respond. However, configuring the optimistic approach in a way that discourage misbehavior could be challenging (incentive-compatibility for verifiers, handling disputes objectively, etc.). This is in addition to the long response finality issue discussed before.

Trusted hardware. TEEs simplify system design and promote performance—as reported in Town Crier [22], latency for different applications were under 600ms. Also, they promote accuracy (as the TEE is trusted to follow the protocol) and compatibility (authenticity boils down to verifying the TEE's signature). However, these come at the expense of strong trust assumptions, that may not hold in practice, as discussed before.

No trust beyond the data source and a third-party verifier. This is usually offered by TLS oracles using a third-party verifiable TLS (and ZKPs for private data). However, they support only URL queries and assume a trusted (or non-colluding) third-party verifier.

Q5: How about overhead? Developers must consider oracle usage overhead, in terms of both fees and gas (or on-chain processing) cost, when evaluating cost efficiency of oracles.

As discussed in Section 6, fees vary across query types and blockchains, and oracles implement different fee structures. For instance, Pyth pull-based data feeds charge 0.000003 ETH per query on Ethereum versus 0.1 POL on Polygon—approximately \$0.0069350 and \$0.009209 USD (using prices from April 2026), respectively [102]. Also, for Chainlink's arbitrary queries and off-chain computations, a USD-denominated premium of \$3.2 is charged per-request [103]. In contrast, immediate-read feeds are priced based on off-chain agreements between the oracle and dApps; information that is not usually public, which highlights a challenge to estimating an oracle's cost beforehand.

Oracle data that require explicit requests increases the gas cost of dApps. In practice this cost will vary oracle to oracle for the same data using the same query type. Similarly, querying the same data across query type may result in different cost. for the same data Consequently, this non-unifirom trend requires granular cost evaluation of oracles, and even query types within the same oracle.

Q6: How about developer experience? Complete testing of dApp contracts is essential before production deployment. Testnets may not support all oracle capabilities offered on the mainnet. Also, some oracles require manual configuration by their development team to enable testing (as in Tellor). Such factors may slow down, or even hinder, full testing of a dApp functionality.

Even for use on mainnet, external tooling is typically provided to facilitate arbitrary URL and computation queries. For example, Chainlink offers a dedicated development tool-chain [104] and Witnet provides an SDK [105] to help aid with query formulation. While helpful, such tooling introduces new maintenance burdens, e.g., Witnet's SDK relies on the now-deprecated Truffle development suite [106] leaving it non-functional until an on-going migration is completed. Furthermore, some feeds may require establishing off-chain agreements with oracles, or even new partnerships

with particular data providers for new feeds. All these factors further impact developers' experience while building their dApps.

8. Concluding Remarks and The Road Ahead

We discuss 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. This may lead to various security issues (we report security incidents around oracles in Appendix A). 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) [107], or the Interactive Universal Composability (iUC) framework [108] that prior work [109] 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 [110], [111] or collaborative ZKPs [112]. 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 [113] supports off-chain MPC and Oasis Sapphire [114] 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) [115], [116]. 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 [117], [118] 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 [119]. 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. Decentralized block building markets [120] have reshaped transaction settlement. They separate party roles into a searcher (who identifies value-extracting strategies), a block-builder (who package transactions into candidate blocks) and a proposer (a validator with proposal rights). Protocols such as MEV-Boost [121] and MEV-blocker [122] adopt this design to redirect profits from searchers and builders to proposers while improving transaction fairness. Moreover, protocols like MEV-Share [123] extend this mission to redistribute a fraction of the collected MEV among 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 [124], Pyth's express relay [125], Chainlink's SVR [126], and API3's OEV Network [127] 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 [128]). Private RPC endpoints, like MEVBlocker [122] and Flashbot's Protect [129], have users send transactions directly to a single trusted block-builder to avoid exposure to the public mempool. Finally, batched threshold encryption techniques [130], [131] 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. The variety in oracles' designs complicates studying OEV. For example, oracles that operate app-chains and employ blockchain messaging protocols introduce new layers of OEV. These 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. There is still a need for large-scale OEV-specific measurement studies to unravel the impact of such factors, and to examine the effectiveness of currently-deployed application-specific redistribution mechanisms.

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 transactions within a block, or block building markets that 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 [132] that outlines a holistic system for programmable-privacy. Another direction is exploring partial privacy, where 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 [133], [134], major new agencies [135], and even government agencies [136], [137] 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 [138], 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 [139]. Similar pilots with Revolut [134], Sygum Bank and Fidelity International [133], 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. It also points to 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 [140] has explored this space, discussing how such ML-assisted models tend to improve existing statistical aggregation approaches in both benign [141] (preventing inaccurate or noisy data being sent downstream) and malicious ones [142], [143] (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 while preserving security.

LLMs and Agentic AI. Integrating AI agents with blockchains [144] 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 pre-defined 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 [140]). On the flip side, oracles and blockchains can be used as a complementary tool to LLM [145], 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 [146] (who incorporates the x402 payment standard [147] into conventional APIs), providing on-chain payment rails for agents. This 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 [148], and resolve ambiguous real-world events for prediction mar-

kets [149].[3] Also, Supra is currently developing multi-agent AI oracles (i.e., aggregation from multiple LLMs) [150] and Gora recently announced efforts in developing AI oracles [151].

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 [152], and prompt bias [153]–[155], reintroducing the core oracle problem of determining an agreed-upon truth. Aggregation and consensus based approaches can be utilized among multiple agents,[4] but the additional overhead and lack of cryptographic verifiability without additional off-chain infrastructure [140] 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 [156]–[158]. 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. CNS2226932.

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," [n. d.], https://synthetix.io/.

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

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

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

3. This has been used in Polymarket [61], 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.

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

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

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

[13] "Tlsnotary - a mechanism for independently audited https sessions," 2014, 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)," [n. d.], https://web.archive.org/web/20260000000000*/https://app.provable.xyz/home/features.

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

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

[19] "UMA data verification mechanism: Adding economic guarantees to blockchain oracles," 2022, 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," [n. d.], https://docs.makerdao.com/smart-contract-modules/oracle-module.

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

[26] "Zkpass," 2026, https://zkpass.org/.

[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," 2026, 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://web.archive.org/web/20260000000000*/https://dos.network/.

[35] "API3: Decentralized APIs for Web 3.0," 2020, https://github.com/api3dao/api3-whitepaper/blob/master/api3-whitepaper.pdf.

[36] "Witnet," 2026, 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] 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.

[44] "Nest protocol," https://www.nestprotocol.org/.

[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] A. Pupyshev, D. Gubanov, E. Dzhafarov, I. Sapranidi, I. Kardanov, V. Zhuravlev, S. Khalilov, M. Jansen, S. Laureyssens, I. Pavlov *et al.*, "Gravity: a blockchain-agnostic cross-chain communication and data oracles protocol," *arXiv preprint arXiv:2007.00966*, 2020.

[47] G. Wood *et al.*, "Polkadot: Vision for a heterogeneous multi-chain framework," *White paper*, vol. 21, no. 2327, p. 4662, 2016.

[48] A. Pasdar, Z. Dong, and Y. C. Lee, "Blockchain oracle design patterns," *arXiv preprint arXiv:2106.09349*, 2021.

[49] 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.

[50] 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.

[51] "Uniswap," [n. d.], https://app.uniswap.org/.

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

[54] "Charli3," [n. d.], https://charli3.io/.

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

[56] 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*, 2019.

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

[58] 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.

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

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

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

[62] Decrypt, "Polymarket rules 'No' on $237M controversial bet over Zelenskyy's suit," Jul. 2025, https://decrypt.co/329210/polymarket-rules-no-237m-bet-zelenskyys.

[63] J. Wen, J. Zhou, and J. Huang, "Can llms help decentralized dispute arbitration? a case study of uma-resolved markets on polymarket," *arXiv preprint arXiv:2604.15674*, 2026.

[64] "IPFS," https://ipfs.tech/.

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

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

[67] 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.

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

[69] (2025) Introducing sp1-cc: Unleashing the full power of the evm. [Online]. Available: https://blog.succinct.xyz/sp1-cc/

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

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

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

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

[74] "Chainlink data stream launches," 2024, https://www.prnewswire.com/news-releases/chainlink-data-streams-launches-on-mainnet-as-new-high-speed-oracle-solution-for-defi-301944062.html.

[75] "Pyth's next chapter: Infrastructure upgrade and a revenue-based economic model," Pyth Network, apr 2026, https://www.pyth.network/blog/pyth-s-next-chapter-infrastructure-upgrade-and-a-revenue-based-economic-model.

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

[77] S. A. Williams, V. Diordiiev, and L. Berman, "Arweave: A protocol for economically sustainable information permanence," 2019, https://www.arweave.org/yellow-paper.pdf.

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

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

[80] 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.

[81] 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.

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

[83] 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*, 2019.

[84] 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.

[85] 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.

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

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

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

[89] 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.

[90] "Saftey net depreciation," 2026, https://developer.android.com/training/safetynet/deprecation-timeline.

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

[92] 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.

[93] 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.

[94] "Tellor layer," 2023, https://tellor.io/blog/tellor-layer/.

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

[96] "Wormhole," 2026, https://wormhole.com/.

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

[98] Polymarket, "How are prediction markets resolved?" 2025, https://help.polymarket.com/en/articles/13364518-how-are-prediction-markets-resolved.

[99] "Chainlink - data feeds," 2026, https://data.chain.link/feeds.

[100] Chainlink, "Data sources," 2026, https://docs.chain.link/data-feeds/data-sources.

[101] Band Protocol, "Band price feeds," https://mainnet-feeds.bandprotocol.com/.

[102] Pyth Network, "Current fees," https://docs.pyth.network/price-feeds/core/current-fees.

[103] Chainlink, "Chainlink functions billing," [n. d.], https://docs.chain.link/chainlink-functions/resources/billing.

[104] "Chainlink Functions Toolkit," 2024, https://www.npmjs.com/package/@chainlink/functions-toolkit.

[105] "Witnet- witnet solidity," 2023, https://github.com/witnet/witnet-solidity.

[106] "Consensys announces the sunset of truffle and ganache." 2023, https://consensys.io/blog/consensys-announces-the-sunset-of-truffle-and-ganache-and-new-hardhat.

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

[108] 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.

[109] 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.

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

[111] 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*, 2025.

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

[113] "Partisia," 2026, https://partisiablockchain.com/.

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

[115] 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.

[116] 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*, 2021.

[117] D. Digital, "Api3: The state of oev," 2024, https://members.delphidigital.io/reports/api3-the-state-of-oev.

[118] 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.

[119] "The dai stablecoin system," 2017, https://makerdao.com/whitepaper/DaiDec17WP.pdf.

[120] V. Buterin, "Proposer–builder separation: Friendly fee market designs," 2021, https://ethresear.ch/t/proposer-block-builder-separation-friendly-fee-market-designs/9725/.

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

[122] "Mevblocker," [n. d.], https://mevblocker.io/.

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

[124] "Oval: Oracle value aggregation layer," 2023, https://github.com/UMAprotocol/oval-contracts.

[125] Pyth, "Express relay: Priority auctions for efficient defi markets," 2024, https://www.pyth.network/blog/express-relay-priority-auctions.

[126] Chainlink, "Introducing smart value recapture (svr): A chainlink-powered mev recapture solution for defi," 2024, https://blog.chain.link/chainlink-smart-value-recapture-svr/.

[127] API3, "Oev network – in depth," https://docs.api3.org/oev/.

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

[129] Flashbot, "Flashbot's protect rpc," 2026, https://docs.flashbots.net/flashbots-protect/overview.

[130] 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.

[131] 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.

[132] Flashbots, "The future of mev is suave," 2022, https://writings.flashbots.net/the-future-of-mev-is-suave.

[133] "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/, 2024.

[134] "Revolut's digital asset market data integrated into the Pyth Network," https://financialit.net/news/blockchain/revoluts-digital-asset-market-data-integrated-pyth-network, 2025.

[135] "Ap and chainlink partnership," https://www.ap.org/media-center/press-releases/2021/ap-chainlink-to-bring-trusted-data-onto-leading-blockchains/, 2021.

[136] "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/, 2025.

[137] "A new wave of economic data is now onchain," https://www.pyth.network/blog/a-new-wave-of-economic-data-is-now-onchain, 2025.

[138] "Chainlink's work with major banking and capital markets institutions," https://blog.chain.link/chainlinks-work-with-major-banking-and-capital-markets-institutions/, 2025.

[139] "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/, 2025.

[140] G. Caldarelli, "Can artificial intelligence solve the blockchain oracle problem? unpacking the challenges and possibilities," *Frontiers in Blockchain*, vol. 8, p. 1682623, 2025.

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

[142] 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.

[143] 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," *IEEE Transactions on Services Computing*, 2025.

[144] N. Romandini, C. Mazzocca, K. Otsuki, and R. Montanari, "Sok: Security and privacy of ai agents for blockchain," in *International Conference on Blockchain Computing and Applications*, 2025.

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

[146] "Ready your apps for agentic commerce," https://corbits.dev/.

[147] "x402 - an open protocol for internet-native payments," https://www.x402.org/.

[148] "Transforming asset servicing with ai, oracles, and blockchains," 2024, https://pages.chain.link/hubfs/e/transforming-asset-servicing.pdf.

[149] "Empirical evidence in ai oracle development," 2025, https://blog.chain.link/ai-oracles/.

[150] Supra, "Threshold ai oracles: Verified ai for event-driven web3," 2025, https://supra.com/documents/Threshold_AI_Oracles_Supra.pdf.

[151] Gora Network, "The network built for agents," https://www.gora.io/blog-post/gora-network-for-agents, 2026, published April 20, 2026.

[152] 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.

[153] 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.

[154] 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.

[155] 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.

[156] 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.

[157] 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.

[158] Anthropic, "A small number of samples can poison llms of any size," https://www.anthropic.com/research/small-samples-poison, 2025.

[159] Synthetix, "Response to Oracle incident," Jun. 2019, https://blog.synthetix.io/response-to-oracle-incident/.

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

[161] M. Nadler, K. Schuler, and F. Schär, "Blockchain price oracles: Accuracy and violation recovery," *Journal of Corporate Finance*, p. 102908, 2025.

[162] 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*, 2025.

[163] Solidus Labs, "The mango markets exploit: An order book analysis," 2022, https://www.soliduslabs.com/post/mango-hack.

[164] Harvest Finance, "Harvest flashloan economic attack post-mortem," Oct. 2020, https://medium.com/harvest-finance/harvest-flashloan-economic-attack-post-mortem-3cf900d65217.

[165] Chainalysis Team, "Oracle manipulation attacks rising: A unique concern for DeFi," Mar. 2023, https://www.chainalysis.com/blog/oracle-manipulation-attacks-rising/.

[166] samczsun, "So you want to use a price oracle," Nov. 2020, https://www.paradigm.xyz/2020/11/so-you-want-to-use-a-price-oracle.

Appendix

There have been a range of security incidents associated to oracles, with a large fraction involving price manipulation that led to significant downstream losses in DeFi applications. In understanding these, it is important to distinguish between the different incident types and the oracle type implicated in each.

Ambiguity in human-mediated oracles and governance-based resolution. As mentioned before, human-formulated queries and responses may introduce ambiguities and subsequent human-mediated or governance-driven resolution processes. Namely, large stakeholders may steer the resolution towards a profitable outcome regardless of the ground truth. Some of these incidents have been witnessed recently on Polymarket [61], an event-based prediction market that utilize UMA. Disputes are handled by UMA token holders via a stake-based weighted voting. A prominent example arose in July 2025 [62], [63] that attracted over 237 million USD in trading volume. The event had been resolved as "NO" by UMA token holders despite widespread media coverage identifying the correct answer to be yes.[5] Such incidents point to the challenges tied to human-formulated queries as well as optimistic approaches; it is hard to enforce one meaning of natural language queries, and financial security collapses when the downstream gains sufficiently outweigh the value of truthful responses and/or handling disputes in an objective way.

Data feeder failures. Here, data feeders fail to report data retrieved from real-world data sources correctly. A notable incident under this category involves an internal oracle used by Synthetix for the Korean Won (sKRW) exchange rate in 2019 [159]. In particular, one of the data feeders reported the exchange rate at approximately $1000\times$ its true value. Although, Syntehtix utilized multi-source aggregation, due to an unreported/undisclosed concurrent incident, only two of the skRW data feeders were active at the time. Thus, misreported price had a large impact on the aggregated value. Over 1 billion USD in profits were executed before the Synthetix team intervened and negotiated a return of

5. We refrain from describing the incident to avoid any political implications. There is also another ambiguity dispute case on Polymarket around some recent political events, which we refrain from citing.

funds in exchange for a bug bounty. This incident points to the fragility of aggregation-based oracle design, and the necessity of feeders' liveness monitoring.

Price discrepancies. Several works have examined price deviations introduced in the oracle data pipeline [160]–[162]. They showed empirical analysis finding economically significantly pricing discrepancies occurring across this pipeline for a number of data feeds. Albeit these findings, these works do not explicitly attribute such deviations to adversarial price manipulation, instead discussing a number of other factors that may have led to such scenarios, e.g., fast-changing market conditions and market stress. Nonetheless, these findings highlight that despite the absence of clear adversarial manipulations, oracles' designs may lead to price discrepancies that dApps must account for.

On-chain price manipulation attacks—no oracles. A substantial portion of incidents labeled as "oracle or price feed manipulation attacks" do not in fact involve oracle systems of the kind studied in this work. Instead, these protocols derive a value of some asset directly from on-chain state. Here, the oracle simply refers to deriving a price from another on-chain contract state, typically a spot price value from an automated market maker liquidity pool or internal order book. Thus, price manipulation attacks here result from manipulating protocol's pricing logic rather than an external data pipeline. Due to prevalence in literature and surface-level resemblance to oracle attacks, we briefly discuss such incidents.

In a nutshell, on-chain price manipulation attacks happen when an adversary exploiting a liquidity pool—often via flash loans or a coordinated attack on a low-liquidity pool—to temporarily distort the interpreted price read by downstream pricing logic. This in turn enables opportunities such as arbitrage, premature liquidations, and under-collateralized borrowing. To date, a large number of such attacks have been reported, e.g., [163]–[166]. A Chainalysis report [165] estimated that in 2022 alone DeFi protocols collectively lost over $\$403.2$ millions across 41 incidents of this type.