Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges

Article *in* IEEE Access · January 2020

DOI: 10.1109/ACCESS.2020.2992698

4 authors:

Hamda Al Breiki
Zayed University
6 PUBLICATIONS **354** CITATIONS
SEE PROFILE

Muhammad Habib ur Rehman
King's College London
74 PUBLICATIONS **4,815** CITATIONS
SEE PROFILE

Khaled Salah
Khalifa University
394 PUBLICATIONS **16,790** CITATIONS
SEE PROFILE

Davor Svetinovic
Khalifa University
128 PUBLICATIONS **3,770** CITATIONS
SEE PROFILE

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Trustworthy Blockchain Oracles: Review, Comparison, and Open Research Challenges

HAMDA ALBREIKI [1], MUHAMMAD HABIB UR REHMAN[2], KHALED SALAH[1,2], AND DAVOR SVETINOVIC[1,2].

[1]Electrical Engineering and Computer Science, Khalifa University of Science and Technology, Abu Dhabi, UAE
[2]Center for Cyber-Physical Systems, Khalifa University of Science and Technology, Abu Dhabi, UAE

Corresponding author: Davor Svetinovic (e-mail: davor.svetinovic@ku.ac.ae).

ABSTRACT The essence of blockchain smart contracts lies in the execution of business logic code in a decentralized architecture in which the execution outcomes are trusted and agreed upon by all the executing nodes. Despite the decentralized and trustless architectures of the blockchain systems, smart contracts on their own cannot access data from the external world. Instead, smart contracts interact with off-chain external data sources, called oracles, whose primary job is to collect and provide data feeds and input to smart contracts. However, there is always risk of oracles providing corrupt, malicious, or inaccurate data. In this paper, we analyze and present the notion of trust in the oracles used in blockchain ecosystems. We analyze and compare trust-enabling features of the leading blockchain oracle approaches, techniques, and platforms. Moreover, we discuss open research challenges that should be addressed to ensure secure and trustworthy blockchain oracles.

INDEX TERMS blockchain, data attestation, decentralization, oracles, smart contract, trust

I. INTRODUCTION

BITCOIN was the first cryptocurrency that eliminated third party by introducing a new decentralized computational architecture in the financial sector. All Bitcoin transactions are stored in a blockchain, which is a distributed public ledger of all transactions or digital events that have been executed and shared among the network participants and it was first used with Bitcoin [1]. Today blockchain applications go far beyond the scope of cryptocurrencies. Modern blockchains enable smart contracts which allow communicating parties to establish agreements based on predefined rules and without the need for a trusted third party. Various possible applications for smart contracts have been explored including healthcare, trading, transportation, IoT, digital rights managements and governmental services [2], [3].

Smart contracts need to acquire data about real-world state and events, from outside the blockchain system [4], [5], which cannot be achieved by smart contracts because blockchain environment is isolated from the external world [6]. To overcome this limitation with smart contracts, there has been a need for data feeds to bring external data into the blockchain system. These data feeds are known as oracles [7]–[9]. The oracles are represented by smart contracts on the blockchain that serve data requests by other smart contracts [4]. Although oracles bring external data on the blockchain, it is hard to provide the guarantee on the trustworthiness of the external data sources [10]–[13]. The use of oracle brings back the centralization problem to the blockchain, since relying on a single source of input not only negates the decentralization principle but also leads towards the risk of bringing corrupt, malicious, and incorrect data on the blockchain. The centralization of oracle causes *"The Oracle Problem"* which brings the dilemma between efficiency and decentralization when oracles fetch real-world events data from external data sources [14].

Considering the essential need of trustworthy oracles for the future blockchain applications, different proposals, early implementations, and platforms have emerged recently [15]–[18]. However, a few studies considered the issues of trust and reliability on blockchain oracles [19], [20]. For example, researchers studied the notion of trust and proposed

a framework to perform the reliability analysis of different blockchain oracle platforms [19]. Similarly, researchers in [20] proposed a decentralized oracle to verify the interactions and resolve the disputes between blockchain smart contracts and oracles. The issue of trust on blockchain oracle was further explored by researchers in [21] whereby they studied different trust factors to embed their proposed trust model in fog computing systems. Researchers in [22] studied trust on blockchain oracles in terms of on-chain data interactions. However, to the best of our knowledge there still exists a significant need to develop the taxonomy and review the existing state-of-the-art platforms.

The primary contributions of this paper are:

- We articulate a concise taxonomic discussion on blockchain oracles that employ centralized and decentralized trust models and design patterns to enable inbound and outbound interactions between smart contracts and various types of oracles.
- We conduct an exhaustive search and present a detailed review and comparison of state-of-the-art blockchain oracle platforms and early implementations/proposals.
- We highlight and discuss pivotal open research challenges to enable trustworthy blockchain oracles.

The paper is organized as follows: Section II discusses the blockchain oracles and the notion of trust in them. Section III presents the taxonomy of blockchain oracles. Section IV presents the review and comparison of the early blockchain oracle implementations. Section V presents the future research challenges. Section VI presents the conclusions.

II. TRUST IN BLOCKCHAIN ORACLES

Blockchain solves the problem of trust in decentralized distributed systems. It combines different existing technologies such as distributed systems for data storage, peer-to-peer protocols for decentralized communication, cryptographic algorithms for security and privacy, and consensus mechanisms for decentralized governance. These integrated technologies ensure an immutable, distributed, always available, and publicly accessible data ledgers. Consequently, blockchain enables consensus-based agreement on a particular state of a digital asset and it ensures permanent, secure, and verifiable storage of agreed data without the need for a third-party.

There is a common misinterpretation about blockchain and trust, where people assume that blockchain networks are “trustless” environments [23]. However, blockchain does not eliminate trust, but it changes the form of trust. For example, the roles of traditional trusted intermediaries and centralized governing bodies are being replaced with sophisticated algorithms while the interpersonal trust between transacting parties is evolving in the form of smart contracts. The blockchain distributes trust among different actors (also known as participants) in the system and it provides various economic models to incentivize the actors to adhere the rules for participation.

Blockchain protocol employs incentive structures predicated on game theory mechanisms to encourage the players (users and miners) in the system to act honestly. The users in this system do not trust each other, and they do not know each other. The objective for those players is to increase their chance to receive the incentive of mining new blocks in a trustless decentralized system. This new paradigm shift for trust was introduced in 2008 when Satoshi Nakamoto published a paper that discussed the need for an electronic payment system to solve the weakness of a trust-based payment model that relies on a trusted third party and replace it with a cryptographic proof-based system. Nakamoto proposed a public blockchain-based decentralized payment system (Bitcoin) for financial transactions without relying on trust and he incorporated incentive models from game theory to encourage nodes to stay honest while using the system [24].

Blockchain introduced a novel approach to building a more competent, secure Public Key Infrastructure (PKI) that helped to overcome some of the existing challenges with current PKI systems. The current implementation of public-key cryptography requires PKI and the existence of a central trusted authority known as Certificate Authority (CA) [25]. With this central authority, many security concerns arise, such as a singular point of failure, integrity, and attacks. Blockchain came to eliminate the need for this type of central trusted party. Blockchain eliminates the need for PKI and CA and all the key management related issues are handled by the blockchain implementation in a decentralized way whereby blockchain acts as a decentralized key-value storage system. Blockchain is capable of securing the data to prevent man-in-the-middle attacks and to minimize the power of third parties. The decentralized key management can tackle the problems with the CA systems through certificate revocation, eliminating single points of failure, and reacting fast to misuses of CAs.

Blockchain is sometimes described as "trust-free technology", which means blockchain should automatically record transactions in the form of publicly immutable data records which are governed by the whole system and it should mitigate the trust issues between peer nodes on the network. However, blockchain technology in itself does not guarantee the trust-free system without relying on external trusted actors. Hence, it is needed to remove the trust-barrier considering the internal complexities of blockchain systems and the trust requirements brought by external actors on the network [26]. For example, smart contract based sharing economy applications cannot be considered as the fully self-contained and highly trusted ecosystem as they rely on both internally generated transactions and the real-world data in the external environments, therefore it is needed to integrate the "trust interface" between blockchains and external shar-

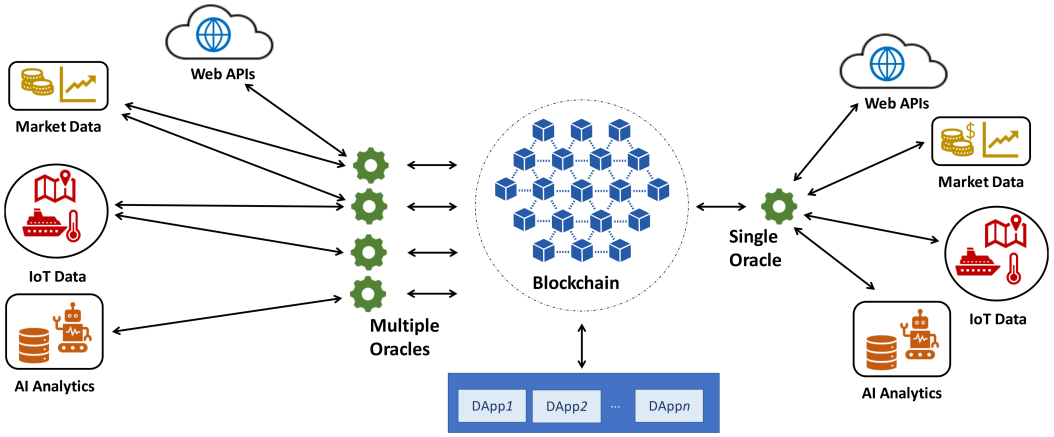

FIGURE 1: The role of oracles in blockchain ecosystems

ing economy platforms [26].

Cryptocurrencies are representative of the first phase of blockchain "Blockchain 1.0", whereas smart contracts are the main advancement of the second phase of blockchain "Blockchain 2.0". The idea of smart contracts was first introduced in 1994 by Nick Szabo, who envisioned smart contract as an automated transaction protocol to execute the contract terms, to minimize the intermediary role of trusted third-parties between transacting users, and to reduce the occurrences of malicious or incidental exceptions. However, the concept of smart contract was first implemented in the context of blockchain technologies. Nowadays smart contracts extend the applications of blockchain technologies to different industries and enabled them to collaborate with each other.

Morabito divided smart contracts into two types: deterministic and non-deterministic smart contracts [27]. As the name suggests the deterministic smart contract code is executed in isolation of external environments and the contract state is maintained and determined by actors inside the blockchain systems. Alternately, the non-deterministic smart contract code requires external information to make decisions which increase its dependability on the actors outside the blockchain systems. The external actor, for instance, a weather information service provider or a sensor data provider, are mapped on to the blockchain in the form of oracle contracts.

Back in ancient times, an oracle was considered to be someone who brings advice or prophesy thought directly from a divine source, however, in modern days, any good source of authentic and reliable information can be considered an oracle. In computer science, oracles are known for their ability to bring reliable external information from outside a system that is not directly accessible from inside the system [27]. However, an oracle, in the blockchain context is an external data agent that observes the real-world events and reports them back to the blockchain to be used by smart contracts. As mentioned earlier, oracles are needed for non-deterministic smart contracts [27] and they are important to integrate smart contracts with the real world. The provided data by oracles may be publicly available, such as asset prices or information on world events, or it may be private, such as bank account information or identity verification. Figure 1 illustrates a general overview of the role of oracles in blockchain systems, where oracles connect external world data to blockchain in both centralized and decentralized topological settings.

The blockchain-oracle workflow is typically executed among three types of participants i.e. 1) data feed providers, 2) oracle nodes/network operators, and 3) blockchain operators. The data feed providers enable different Web APIs and communication interfaces to read and provide data from various online data sources such as sensors, stock markets, crypto-exchanges, and web-enabled ERPs to oracle nodes. The oracle node operators enable notarization and certification policies to transfer highly accurate, relevant, and reliable data to blockchain systems. Finally, the blockchain operators enable the ecosystems to securely and safely execute the smart contracts on underlying peer to peer networks. The issues of trust may arise due to any of the participants. For example, the data feed providers may continuously report the corrupt or malicious data or the data feeds may get compromised where the adversarial attacks could be penetrated into oracles to alter their notarization and certification policies. Similarly, adversaries could attack through oracles to manipulate the data feeds to get favorable reputations from the blockchains. The adversaries could also attack the blockchain networks to profit from their consensual processes and benefit from their mining algorithms. Therefore, the notion of trust in blockchain oracles, in a broader sense, encapsulates the activities of all the participants on the network. Hence the trust models should comply with the security, privacy, reliability, authenticity, and reputation of all the stakeholders.

III. ORACLES TAXONOMY

Figure 2 shows categorization of oracles based on their data sources, number of nodes, design patterns or their interaction models.

A. DATA SOURCE

Oracles can get data from various sources, and based on these sources following classification can be defined:

Software oracles deal with data originated from online sources on the Internet, browsing through these data to find the information it needs, extracting the required information and returning it to the smart contract. A few examples of data that software oracles can gather are gold price, flight delays, and currency exchange rates.

Hardware oracles gather data directly from the physical world by means of things such as scanners and sensors. A few examples for hardware oracles include RFID sensors to track supply chain goods, and temperature and humidity sensors in shipment containers.

Human oracles rely on people's actions to provide external data to blockchain systems. Human oracles provide smart contracts with answers to questions. For example, people can vote on the truth of an event.

B. TRUST MODEL

The number of nodes used by oracles to get data to smart contracts defines the trust model used by oracles.

Centralized trust model relies on data from a single source. The efficiency in centralized trust model is high but it represents a single point of failure, where availability, accessibility, and level of certainty about the validity of the data depends only on one node.

Decentralized trust model resolves the singular point of failure problem in the centralized trust model. However, these models bring higher latency for data processing with less efficiency, when compared with the centralized trust model.

C. DESIGN PATTERN

The set up of an oracle can have primarily three forms as follows:

Request-response design pattern is used when the data space is too huge to be stored in a smart contract and users are expected to only need a small part of the overall dataset at a time. An applicable use case for this design pattern is for data providers. This design pattern can be implemented as a system of on-chain smart contracts where the requests to the oracle are initiated and off-chain infrastructure is used to monitor requests, retrieve and return data.

Publish-subscribe design pattern is used for an oracle that effectively provides a broadcast service for data that is expected to change like price feeds and weather information. This pattern is similar to RSS feeds where the oracle is updated with new information and a flag indicates that new data is available for subscribers. Subscribers can either poll the oracle to check whether the latest information has changed or listen for updates to oracle contracts and act when they occur.

Immediate-read design pattern is used for oracles that provide data which is only needed for an immediate decision like academic certificates and dial codes. This type of oracle stores data once in its contract storage and it can be updated. The data in the oracle's storage is available for any other smart contract to look it up using a request call to the oracle contract, and also the data available for direct look-up by blockchain-enabled applications.

D. INTERACTION

Oracles can have different interactions with the external world, by either insert data to the blockchain or deliver data from blockchain to the external world. Based on the type of interaction, oracles can have two types:

Inbound Oracles insert data from the external world into the blockchain. An example would be the price of an asset, which can be purchased automatically when it reaches the desired price.

Outbound oracles allow smart contracts to deliver data to the external world. The smart lock is an example of outbound oracles when payment is received on the blockchain address of the smart lock, the smart lock in the physical world will unlock automatically.

IV. TOWARDS TRUSTED ORACLES

Multiple academic and industry research works implemented trust models for blockchain oracles.

Provable, (previously known as Oraclize), is a pioneer oracle service (operating since 2015) that provides safe data-transport-layer for smart contracts to fetch external data from Web APIs [28]. The main objective of Provable is to ensure the availability of verifiable and auditable off-chain computation archives. The oraclize engine in Provable enables the existing Web API for any external off-chain data sources and store the data with authenticity proofs using decentralized storage mechanisms such as IPFS and SWARM. Provable is using a variety of authenticity proofs to verify the genuineness of the data fetched from original sources and that the data is untampered. The model used by Provable's oracle, as depicted in Fig 3, uses Trusted Execution Environments (TEE) and auditable virtual machines to build authenticity proofs (such as TLSNotary). Provable's oraclize engine ensures a synchronous communication between on-chain smart contracts and off-chain external data sources. For any arbitrary interaction, an on-chain smart contract executes two transactions whereby first transaction performs the data call-out request in the form of query whereby the data source and the actual query are specified. Typically, the call-out query specifies a reference of the data to the oraclize engine.

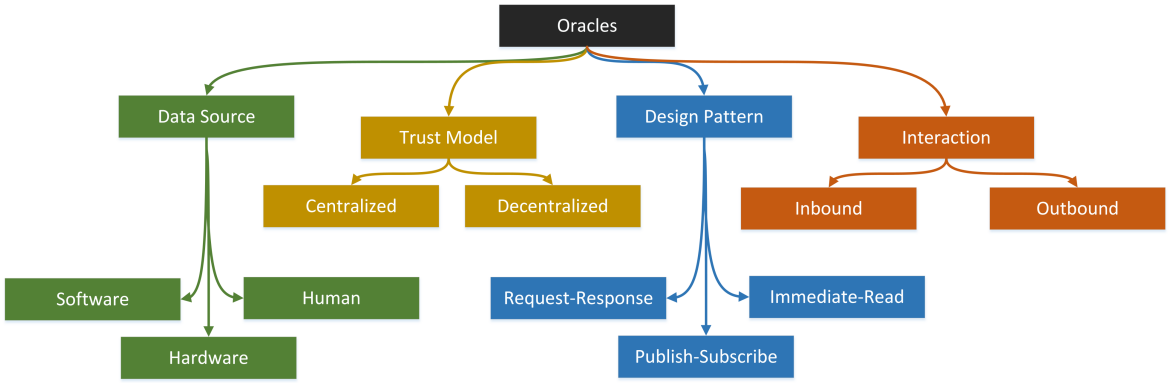

FIGURE 2: Taxonomy of Blockchain Oracles

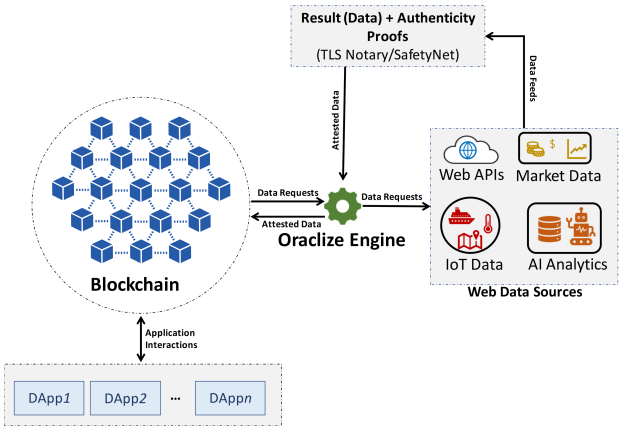

FIGURE 3: System diagram of Provable [28]

The oraclize engine collects and attests requested data using TLSNotary proofs (which are a collection of digital signatures) and returns it to the calling smart contract in the form of a new transaction having call-back method.

Although Provable guarantees the auditable data provisioning, a few limitations of current blockchain systems degrade Provable's performance. The limited functionality of Ethereum Virtual Machines (EVM) such as inefficiency to handle precision-bound floating point numbers, opcodes, or precompiles, the requirement to use minimal gas, high cost of operating blockchain network, inability to reuse existing libraries, absence of confidentiality and privacy, and the scalability are major bottlenecks. Considering the scalability constraints, Provable focuses on non-iterative transaction execution whereby a data request gets a secure and trustworthy response within a single callout-callback cycle. The essence is that oraclize engine provides the security guarantees and proves that a given piece of code has been executed as intended in an off-chain context without needing to execute the code again. The oraclize engine's single execution cycle approach is efficient and cost-effective but it only works well with deterministic callback transactions. However, it benefits in terms of verification with iterative execution and provisioning of fully auditable off-chain oracles.

The execution of TLSNotary certificates in external systems required dependency over trusted third parties. *TownCrier* enables hardware-based TEEs whereby the data authentications are performed in trusted and secure execution environments [4]. TownCrier connects blockchain smart contracts with http-enabled data sources (primarily websites on the Internet) and establishes a secure bridge for data transfer between both endpoints. Fig 4 shows the system architecture of TownCrier platform. The core logic of TownCrier is executed in Intel's Software Guard Extensions (SGX) enclave as trusted code on TownCrier server. The SGX enclaves inherit the blackbox implementation properties of a TEE where neither the operating system nor other applications on the CPU can interfere with the applications running inside TownCrier enclaves. This trusted hardware capability of SGX enables data collection from multiple data feeds, performs secure data aggregation, and adds authenticity proofs before sending it back to blockchain smart contracts. The TownCrier smart contract on the blockchain intermediates between users and SGX enclaves on the TownCrier server in order to ensure the high-availability of trusted data feeds to connected DApps. Overall, In addition with providing security, SGX provides guarantees of integrity and confidentiality. TownCrier preserves the integrity using hashing algorithms which take different measurements relating to application starting time and memory and generate signatures to attest the data feeds. It also ensures the confidentiality of requested data using public-private key-value pairs. TownCrier encrypts the data inside enclaves and shares its corresponding public keys to TownCrier smart contracts on the blockchain, which in turn, transfer the data according to user requirements.

TownCrier enables trusted oracles by supporting datagram requests whereby a datagram essentially represents a concise piece of data which can directly trigger some event or response on a requesting smart contract. The secure execution

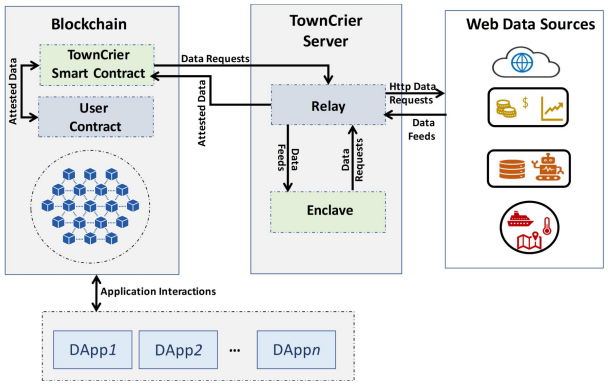

FIGURE 4: System Architecture of TownCrier [4]

of datagrams inside SGX environment relies on implicit trust on SGX based TEE, the TownCrier code, and the validity of requested data at a specified time interval. Towncrier supports the private and custom datagram requests in order to ensure the confidentiality of datagrams as well as their corresponding data feeds. In order to designate private datagrams, TownCrier encrypts the request parameters using public keys, however, it additionally supports the secure and encrypted access-control mechanisms to customize the datagram requests. Hence, it preserves the basic property of datagram authenticity. TownCrier was tested considering universal compatibility framework by achieving a formal model that spans across TEE and blockchain environments. In addition, it complies with two key security properties i.e. 1) *gas-sustainability,* to ensure Ethereum never raise out-of-gas exception error and 2) *trusted computing based code minimization,* to avoid building a new authenticated channel from blockchain to enclaves and to minimize on-chain signature verification code.

Augur, provides a low-cost oracle platform for prediction markets for online trading [29]. Augur leverages the trust decentralization of the blockchain and opens us the prediction markets to the wisdom of the crowds. With Augur oracle, information about the real world can be transferred to the blockchain without a need for a trusted intermediary. The users who have reputation tokens (Augur's native token) can choose the outcomes of Augur's prediction markets, by staking their tokens on the actual observed outcome. Based on their prediction, they receive settlement fees from the markets. The incentive structure on Augur is designed to encourage users to stay honest and report accurate outcomes to maximize their profit.

The market lifetime on Augur platform spans over four phases such as 1) market creation, 2) trading, 3) reporting, and 4) settlement. At the first stage, everyone on the augur platform is allowed to set the event end time and select the designated reporter, however, a single reporter cannot make a final decision about the prediction event hence the community can always interfere to ensure a dispute-free correct reported by a designated reporter. In addition, the market creator selects the resolution source to be used by reporters to determine the event outcomes. The market creator also posts validity and creator bonds, to be materialized with REP (a native token atop Ethereum's ERC777 token) that incentivize market creators to create objective and well-defined events with clearly outlined outcomes. At the second stage, an ordered book of every prediction market is maintained by Augur's trading contracts. At the third stage, Augur's oracle determines the event outcomes for final settlement whereby the correctly reporting reporters are rewarded while incorrect reporters are penalized. The reporting process of Augur's prediction market, which is executed at the third stage, is presented in Fig 5 whereby an Augur market can be in one of seven different states at any arbitrary instance of time. Finally, all the market payments and disputes are settled at the forth stage.

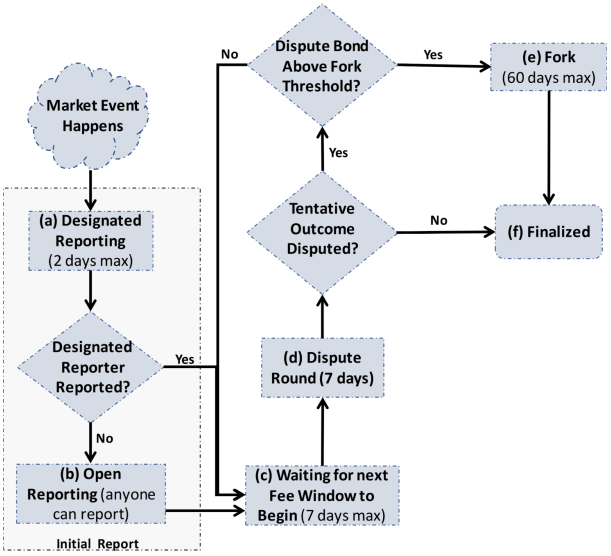

FIGURE 5: Reporting Process at Augur Platform [29]

Augur builds trust and secures oracles by minimizing the profit of successful attack when compared with the cost of launching attack on designated reporters. To achieve this, Augur proposed to slow down the fork time which is almost 60 days in current settings so that new markets could efficiently resolve against the threats of creating new forks. However, Augur's forking protocol still needs to address the issue of parasitic markets, whereby competing markets can offer the similar services without designated reporters.

ChainLink proposed decentralized oracle network in order to enable trustworthy data feeds and connectivity between smart contracts and external data sources [30]. ChainLink high-level architecture, as depicted in Fig 6 distributes the trust models at two layers between Blockchain (on-chain)

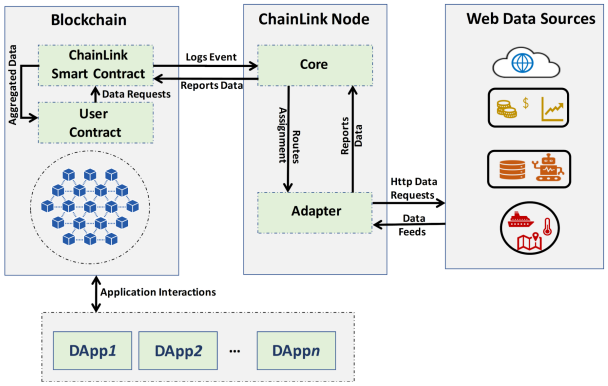

FIGURE 6: Workflow Execution in ChainLink [30]

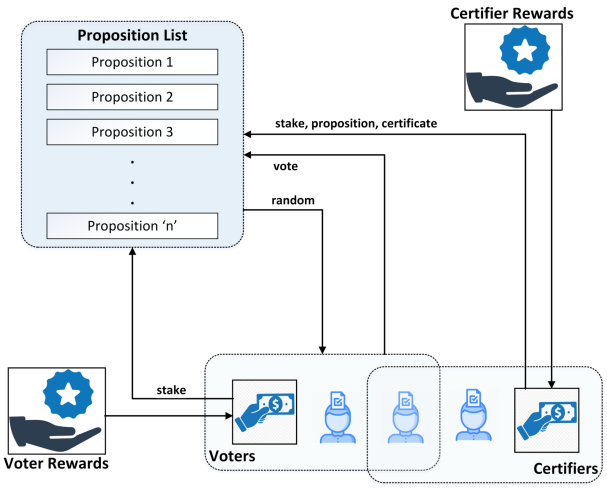

FIGURE 7: High-level System Architecture of ASTREA [31]

and ChainLink Nodes (off-chain). The on-chain components support two types of smart contracts namely 1) user smart contracts to execute the on-chain application logic for decentralized applications and specify the data requirements from external data feeds and 2) ChainLink smart contracts to ensure trust and security via aggregation operations and reputation calculation mechanisms. The off-chain ChainLink Nodes provide two types of components namely 1) ChainLink Core and 2) Adapters. ChainLink Core enables the components to interface with on-chain smart contracts and perform scheduling and balance workloads across various multiple external services. In addition it distributes the tasks and creates assignments whereby each assignment represents a sub-tasks. Furthermore, ChainLink Core routes the assignments to Adapters which are the off-chain representations of external REST-API services.

ChainLink proposed a decentralized and distributed trust model spanning across on-chain and off-chain components which securely pushes the data between smart contracts and Web-API in order to create externally-aware tamperproof smart contracts. This trust model ensures that ChainLink components maintain integrity, confidentiality, and authenticity of data for smart contracts while selecting external oracles, during data reporting sessions between smart contracts and ChainLink Nodes, and aggregating reported query results from multiple data feeds. The trust model also embeds the reputation smart contracts to incentivize and penalize the reporting oracles and maintain fairness among all reporting oracles. The reputation function contains parameterized inputs on the basis of total number of assignments, total number of completed assignments, total number of accepted assignments, average response time, and total amount of penalty paid by the target oracles. The financial transactions on ChainLink platform are performed via Link (developed using ERC20 and ERC223 Ethereum tokens). In addition, ChainLink provides validation system, certification system, and a contract-upgrade service to strictly comply with decentralized design of ChainLink protocol.

ASTRAEA, enables a voting-based game for decentralized blockchain oracles and the entities on ASTRAEA can act as submitters, voters, or certifiers [31]. ASTREA supports external data feeds from multiple oracles and it represents the reported data in the form of Boolean propositions whereby an oracle (e.g. a Web API of a stock prediction market) announces the outcomes of external events which could be validated against their truth or falsehood. Boolean propositions, as shown in Fig 7, are submitted to the system by submitters who allocate money to fund the effort of validating the submitted propositions. A small stake is placed by voters who receive a random proposition and vote on its truth, and they play a low-risk/low-reward game. The certifiers, on the other hand, play a high-risk/high-reward game, where they have the option to choose a proposition and certify the truth of it after placing a large stake. ASTREA incentivizes honest participants for validating propositions. ASTREA's trust model ensures high availability of trusted oracles and it achieves the desirable Nash equilibrium with the assumption that all participants behave honestly on ASTREA's platform [32]. However, the uncontrolled participation from external data feeds can still break the Nash equilibrium by generating off-chain collusion attacks with the help of a selected pool of submitters, voters, and certifiers.

Aeternity is an open-source decentralized application platform that utilizes public blockchain technology. It employs both Proof-of-Work and Proof-of-Stake consensus mechanisms [33]. Considering the fundamental role of blockchain oracles in the future decentralized economy, Aeternity is implementing integrated oracles, which can be used to request and access real-world data from various providers. Aeternity is reusing the consensus mechanism that we use to agree on the state of the system, to also agree on the state of the outside world. Oracle operators in Aeternity can register with the

blockchain, receive questions directed to them, and answer these questions. The questions are posed to an oracle by the smart contracts after paying for posting the question. The smart contract can access both questions and answers.

Aeternity's trust model spans over a complete ecosystem of main-net blockchain, oracles, smart contracts, Aeternity naming system, on-chain and off-chain state-channels, and serialization formats. The Aeternity blockchain main-net enables security by using type-safe virtual machines namely Fast Aeternity Transactions Engine (FATE) which ensures gas efficiency and security. In addition, Aeternity's native smart contract language, called Sophia, and its native cryptocurrency, named as Aeternity token, ensure an inclusive platform with less dependencies over external systems. Aeternity uses Bitcoin-NG as consensus mechanism to enable high transaction throughput when compared to Bitcoin or Ethereum blockchains, however, its support to off-chain state-channels boosts the transaction throughput hence making Aeternity a feasible platform for data-intensive oracles.

PriceGeth, was implemented as a proof of concept to allow a trusted entity to publish price pairs to the blockchain [34]. PriceGeth was implemented as a smart contract to publish real-time price pairs to Ethereum blockchain and it keeps all the historical prices on-chain, so no gas will be required to access the price. PriceGeth workflow, as depicted in Fig 8, involves interactions between Ethereum smart contracts, PriceGeth server, and PriceFetcher module (which fetches stock prices every second from an external Web API). The authors of PriceGeth stated that their design has a central point of failure, and they argued that for decentralized price feeds trust can not be achieved on the blockchain without having decentralized exchange infrastructure. They have also highlighted another challenge in their design, where no incentive is given to price oracles for publishing and storing price pairs, which cost these oracles gas.

Witnet is a reputation-based decentralized oracle network (DON) protocol [35] which enables to connect the smart contract with external data providers. Witnet network runs its native customized blockchain and its own protocol token namely Wit. The data providers on the DON are called witnesses who perform a series of activities termed as Retrieve-Attest-Deliver (RAD) work. Witnesses contribute with their mining power which is mainly determined by their reputation for efficient, honest, and trustworthy execution of RAD jobs on the DON. However, Witnet's reputation system rewards the successful majority consensus witnesses while penalizing the contradicting witnesses. In addition, Witnet enables special kind of participating nodes on the DON, called bridge nodes, which only perform the 'Deliver' part of the RAD work.

A RAD request on the DON is performed in three phases 1) Retrieve phase, to retrieve the information from external data providers using Http requests, 2) Attest phase, to perform consensus on the retrieved information, and 3) Deliver phase, to ensure high availability of consensually attested information to data requesters. In addition, the Witnet participants are categorized as 1) Clients, who generate RAD requests and specify the required number of witnesses and bridges, 2) Witnesses, who execute the retrieve and attest tasks of the RAD requests, and 3) Bridges nodes, who execute the deliver tasks of the RAD requests. However, sometimes these participants play multiple roles whereby bridges represent a subset of witnesses and witnesses further represent a subset of clients. Although Witnet does not employ an orchestrated trust model on its DON, however, it maintains the trustworthiness and honesty of the participants using reputation-based incentives and penalties.

Witnet was designed considering the notion of decidability and verifiability which are the basic requirements to attest the RAD requests on the DON. Although the witnesses and bridges can easily process the decidable requests but it is hard to process the undecidable requests whose truth or falsehood could not be determined within a limited time interval. Witnet enables a special type of RAD requests in order to handle the undecidable request which remain unresolved for an indefinite period of time. Since the the provability or verifiability of RAD request could be determined by the client smart contracts hence Witnet does not deploy any special RAD requests to cater these special needs.

A. COMPARISON OF EXISTING SOLUTIONS

Based on the extensive review, we found that existing blockchain oracle solutions can be differentiated based on different aspects as detailed in Table 1 and Table 2.

The first aspect is related to the deployment of the solution (e.g., whether the solution is deployed on-chain, off-chain, or in both sides). Provable, Town Crier, and PriceGeth provide off-chain solutions that can be connected to a smart contract on-chain to transfer the requested data. Whereas, Witnet, Augur, ASTRAEA, and Aeternity are all on-chain solutions. However, ChainLink provides both on-chain and off-chain components for its solution.

Trust model is the second aspect of analysis. As we discussed in the oracles taxonomy section, the number of nodes used by oracle solution to get data into smart contracts defines the trust model used by oracles, where the single node represents a centralized trust model and multiple nodes represent decentralized models. Three of the above solutions follow a centralized trust model, but each of them is based on a different approach. Provable is an example of a centralized oracle service leveraging a variety of authenticity proofs such as TLSNotary Proof; whereas, TownCrier is another centralized solution that is based on Trusted Execution Environments (TEEs), where it uses Intel's SGX (Software Guard eXtensions) to ensure that responses from HTTPS queries can be verified as authentic. The third centralized solution is PriceGeth, which relies on a single price feed (PriceFetcher). The remaining five solutions rely on multiple nodes to get ex-

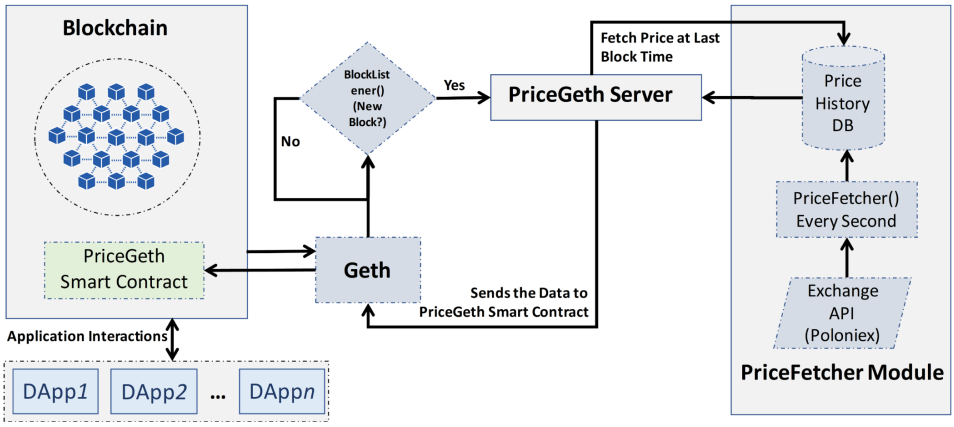

FIGURE 8: PriceGeth Workflow Diagram [34]

TABLE 1: Comparison of existing oracle solutions based upon trust models

Solution	On/Off Chain	Trust Model	Native Token
Provable	Off-chain	Centralized - Authenticity Proofs (TLSNotary)	None
TownCrier	Off-chain	Centralized – TEE (SGX)	None
PriceGeth	Off-chain	Centralized -Single Price Feed	None
Witnet	On-chain	Decentralized - Reputation-Based	Wit
Augur	On-chain	Decentralized - Reputation-Based	REP
ChainLink	Off-chain / On-chain	Decentralized - Reputation-Based	LINK
ASTRAEA	On-chain	Decentralized - Voting-Based	None
Aeternity	On-chain	Decentralized - Consensus-based	Aeon

TABLE 2: Comparison of existing oracle solutions based upon trust features

Solution	Security	Authenticity	Confidentiality	Accessibility
Provable	TEE/TLS Notary	SafetyNet/On-chain Certificates	Public Keys/Web of Trust	Digital Certificates/Public-Private Keys
TownCrier	TEE (SGX)/TLS Notary	Off-chain TLS Certificates	Private Datagram Requests	Custom Datagram Requests
PriceGeth	Smart Contracts	Blockchain Smart Contracts	None	Blockchain and WebAPIs
Witnet	On-Chain	Blockchain Smart Contracts	None	LOCKSS [36]
Augur	Limited Attacker benefits	On-Chain/Reputation	None	None
ChainLink	On-chain Security Services	Digital Signatures	Public-key Encryption	Blockchain and OracleAPIs
ASTRAEA	Blockchain	Voting Based Certification	None	Blockchain and OracleAPIs
Aeternity	Native Language/Secure VMs	On-contract	None	On-chain

ternal data into smart contracts, where these solutions follow a decentralized trust model with different approaches. Witnet is a decentralized oracle network based on reputation points. Augur is another reputation-based decentralized oracle and platform specifically designed for prediction markets. ChainLink operates as a fully decentralized network that is also based on reputation; whereas, ASTRAEA is a decentralized oracle based on a voting game between voters and certifiers and it detects and penalizes the dishonest voters by enabling sealed voting mechanism as well as the assessment of majority votes. However, the possibility of adversarial attacks on oracles still exist whereby an oracle can intercept the responses of other oracles and replicate the voting behavior accordingly to benefit from the voting mechanism. Since oracles on ASTREA can participate as submitters, voters, and certifiers, there still exist the risk of Sybil attacks whereby one oracle may observe the behavior of other oracles and then start acting as certifiers as well. Finally, Aeternity uses state-channels (i.e. off-chain transaction processing) to enable communication between peers on the decentralized network. The blockchain, on the Aeternity, plays the role of arbitrator in case of any dispute between two participants. This trust model ensures more privacy due to less exposure to data on the blockchain. In addition, off-chain state-channels ensures speedy, secure, and low-cost blockchain transactions. Finally in the third aspect, Witnet, Augur, ChainLink and Aeternity use their native tokens; whereas the other solutions do not introduce new tokens.

V. OPEN RESEARCH CHALLENGES

Since the research on trusted oracles is in its infant stages, researchers need to address multiple pressing challenges.

Smart Contract Engineering: The externally imported data on the blockchain could be redundant, manipulated, bogus, and malicious, therefore, the smart contract should provide a mechanism to prevent the entry of unintended data on the blockchain. Besides, the smart contract should provide

a witness mechanism such as reputation algorithms or data forensic strategies to ensure the trustworthiness of external data.

Security and Privacy: The interaction between known and unknown stakeholders on-the-chain and unknown stakeholders off-the-chain could potentially raise the privacy and security concerns at both ends. Therefore, the new mechanisms are required to address the internal (e.g., DAO attacks, re-entry attacks, or DDoS attacks) and external security and privacy issues such as bringing malicious or corrupt data on the blockchain, reputation manipulation attacks, or identity theft attacks.

Responsible Oracles: Since the oracles operate in external environments, there is always a risk of triggering undesired events by oracles which may lead to jeopardizing the performance objectives of underlying decentralized applications. Therefore, new trust-building mechanisms are required to ensure more responsible oracles on the blockchain network. Also, the integration of decentralized identity management and registration services can make oracles more responsible when compared with unregistered oracles.

Design Challenges: The blockchain oracles are always at the risk of centralization, collusion, and Sybil attacks. Therefore, the primary design challenge is to ensure decentralization and reliability, even with fewer oracles. A better approach is to enable multiple oracles reporting the same data streams or events based on on-chain consensus mechanism, however, it will incrementally increase the economic cost of the overall system. The issue of scalability will also arise when blockchain needs to attest to the massive data streams from multiple oracles. The off-chain state-channels or alternate compute-optimal consensus algorithms could be possible solutions to handle the scalability issues.

Implementation Challenge: The smart contracts are mostly written by the open-source community, whereby most of the smart contracts have the unique requirements, but copying the contract code may increase the risk of error-propagation from one smart contract to others which may degrade the performance of blockchain oracles. Efforts are required to build the application or domain-specific communities for blockchain oracles to provide bug-free template smart contracts for open source communities. This requirement also opens the opportunities for startups to provide software-as-a-service smart contracts for each application areas.

External Data Semantics and Governance: Semantic understanding of external data is necessary to ensure consensus base deterministic execution of smart contracts. Furthermore, development of new decentralized governance models are required to govern and manage the data and on-chain interactions between smart contracts and oracles. Efforts are also needed to set new data transmission and data definition standards to ensure reliable communication between oracles and smart contracts. New standards for data translation between multiple blockchain platforms are also desired to ensure fully interoperable blockchain oracle ecosystems.

VI. CONCLUSION

In this paper, we reviewed and discussed trust for oracles when used in blockchain ecosystems. We categorized existing blockchain oracles with respect to data sources, trust models, design patterns, and data interactions between oracles and blockchain platforms. Our discussion and analysis show that achieving a fully trustworthy oracle platform for blockchain is still at an early stage, and more research attention and efforts need to be exerted in this direction. Specifically, we need to integrate novel security and privacy mechanisms into existing trust models, to detect, prevent, and mitigate Sybil and collusion among oracles, to solve design and implementation challenges, and to establish new standards for unified data formats, and decentralized governance.

ACKNOWLEDGMENT

We would like to thank Dr. Claudio Lima from Blockchain Engineering Council, USA for his advice and useful discussions that helped improve this work.

REFERENCES

[1] H. Meng, E. Bian, and C. Tang, "Themis: Towards decentralized escrow of cryptocurrencies without trusted third parties," in 2019 Sixth International Conference on Software Defined Systems (SDS). IEEE, 2019, pp. 266–271.

[2] K. Salah, M. H. U. Rehman, N. Nizamuddin, and A. Al-Fuqaha, "Blockchain for ai: review and open research challenges," IEEE Access, vol. 7, pp. 10 127–10 149, 2019.

[3] M. Nassar, K. Salah, M. H. ur Rehman, and D. Svetinovic, "Blockchain for explainable and trustworthy artificial intelligence," Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 10, no. 1, p. e1340, 2020.

[4] F. Zhang, E. Cecchetti, K. Croman, A. Juels, and E. Shi, "Town crier: An authenticated data feed for smart contracts," in Proceedings of the 2016 aCM sIGSAC conference on computer and communications security. ACM, 2016, pp. 270–282.

[5] K. Yamashita, Y. Nomura, E. Zhou, B. Pi, and S. Jun, "Potential risks of hyperledger fabric smart contracts," in 2019 IEEE International Workshop on Blockchain Oriented Software Engineering (IWBOSE). IEEE, 2019, pp. 1–10.

[6] R. Van Mölken, Blockchain across Oracle: Understand the details and implications of the Blockchain for Oracle developers and customers. Packt Publishing Ltd, 2018.

[7] X. Xu, C. Pautasso, L. Zhu, V. Gramoli, A. Ponomarev, A. B. Tran, and S. Chen, "The blockchain as a software connector," in 2016 13th Working IEEE/IFIP Conference on Software Architecture (WICSA). IEEE, 2016, pp. 182–191.

[8] H. Moudoud, S. Cherkaoui, and L. Khoukhi, "An iot blockchain architecture using oracles and smart contracts: the use-case of a food supply chain," in 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC). IEEE, 2019, pp. 1–6.

[9] H. Albreiki, L. Alqassem, K. Salah, M. Rehman, and D. Svetinovic, "Decentralized access control for iot data using blockchain and trusted oracles," 2019.

[10] P. Szalachowski, "Padva: A blockchain-based tls notary service," in 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, 2019, pp. 836–843.

[11] B. Putz and G. Pernul, "Trust factors and insider threats in permissioned distributed ledgers," in Transactions on Large-Scale Data-and Knowledge-Centered Systems XLII. Springer, 2019, pp. 25–50.

[12] A. Brandoli, "Blockchain notarization: extensions to the opentimestamps protocol," 2019.

[13] S. Gorbunov and H. Wee, "Digital signatures for consensus." IACR Cryptology ePrint Archive, vol. 2019, p. 269, 2019.

[14] Y.-C. Hu, T.-T. Lee, D. Chatzopoulos, and P. Hui, "Analyzing smart contract interactions and contract level state consensus," Concurrency and Computation: Practice and Experience, p. e5228, 2019.

[15] S. Wang, H. Lu, X. Sun, Y. Yuan, and F.-Y. Wang, "A novel blockchain oracle implementation scheme based on application specific knowledge engines," in 2019 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI). IEEE, 2019, pp. 258–262.

[16] A. Schaad, T. Reski, and O. Winzenried, "Integration of a secure physical element as a trusted oracle in a hyperledger blockchain," 2019.

[17] K. Chatterjee, A. K. Goharshady, and A. Pourdamghani, "Probabilistic smart contracts: Secure randomness on the blockchain," in 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC). IEEE, 2019, pp. 403–412.

[18] A. Fujihara, "Proposing a blockchain-based open data platform and its decentralized oracle," in International Conference on Intelligent Networking and Collaborative Systems. Springer, 2019, pp. 190–201.

[19] S. K. Lo, X. Xu, M. Staples, and L. Yao, "Reliability analysis for blockchain oracles," Computers & Electrical Engineering, vol. 83, p. 106582, 2020.

[20] L. Ma, K. Kaneko, S. Sharma, and K. Sakurai, "Reliable decentralized oracle with mechanisms for verification and disputation," in 2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW). IEEE, 2019, pp. 346–352.

[21] P. Kochovski, S. Gec, V. Stankovski, M. Bajec, and P. D. Drobintsev, "Trust management in a blockchain based fog computing platform with trustless smart oracles," Future Generation Computer Systems, vol. 101, pp. 747–759, 2019.

[22] J. Heiss, J. Eberhardt, and S. Tai, "From oracles to trustworthy data on-chaining systems," in 2019 IEEE International Conference on Blockchain (Blockchain). IEEE, 2019, pp. 496–503.

[23] D. Yaga, P. Mell, N. Roby, and K. Scarfone, "Blockchain technology overview," arXiv preprint arXiv:1906.11078, 2019.

[24] S. Nakamoto et al. (2008) Bitcoin: A peer-to-peer electronic cash system.

[25] H. Orman, "Blockchain: The emperors new pki?" IEEE Internet Computing, vol. 22, no. 2, pp. 23–28, 2018.

[26] F. Hawlitschek, B. Notheisen, and T. Teubner, "The limits of trust-free systems: A literature review on blockchain technology and trust in the sharing economy," Electronic commerce research and applications, vol. 29, pp. 50–63, 2018.

[27] V. Morabito, "Business innovation through blockchain," Cham: Springer International Publishing, 2017.

[28] Provable documentation. [Online]. Available: https://docs.provable.xyz

[29] 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.

[30] S. Ellis, A. Juels, and S. Nazarov, "Chainlink: A decentralized oracle network," Retrieved March, vol. 11, p. 2018, 2017.

[31] 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). IEEE, 2018, pp. 1145–1152.

[32] R. Berryhill and A. Veneris, "Astraea: A decentralized blockchain oracle," IEEE Blockchain Technical Briefs, 2019.

[33] Z. Hess, Y. Malahov, and J. Pettersson, "Æternity blockchain," Online]. Available: https://aeternity. com/aeternity-blockchainwhitepaper. pdf, 2017.

[34] S. Eskandari, J. Clark, V. Sundaresan, and M. Adham, "On the feasibility of decentralized derivatives markets," in International Conference on Financial Cryptography and Data Security. Springer, 2017, pp. 553–567.

[35] A. S. de Pedro, D. Levi, and L. I. Cuende, "Witnet: A decentralized oracle network protocol," arXiv preprint arXiv:1711.09756, 2017.

[36] S. University, "Lockss, presrvation priciples," [Online]. Available: https://lockss.org/about/preservation-principles/, 2020.