2019 IEEE International Conference on Blockchain (Blockchain)

# From Oracles to Trustworthy Data On-chaining Systems

Jonathan Heiss, Jacob Eberhardt, Stefan Tai
*Information System Engineering*
*TU Berlin*
Berlin, Germany
{jh,je,st}@ise.tu-berlin.de

***Abstract*—Many blockchain transactions require blockchain-external data sources to provide data. Oracle systems have been proposed as a link between blockchains and blockchain-external resources. However, these Oracle systems vary greatly in assumptions and applicability and each system addresses the challenge of data on-chaining partly. We argue that Data On-chaining must be done in a trustworthy manner and, as a first contribution, define a set of key requirements for Trustworthy Data On-chaining. Further, we provide an in-depth assessment and comparison of state-of-the-art Oracle systems with regards to these requirements. This differentiation pinpoints the need for a uniform understanding of and directions for future research on Trustworthy Data On-chaining.**

***Index Terms*—trustworthiness, on-chaining, oracle, smart contract, truthfulness, blockchain, TLS, Trusted Execution Environment, voting**

## I. Introduction

Blockchains impacted various application domains, ranging from finance or logistics to digital citizenship and human rights. They have important new properties, e.g., immutability and censorship resistance, and enable the decentralized execution of programs called smart contracts.

No single entity is in control, but a group of independent peers share the responsibility of operating the network and processing transactions. Hereby, incentives are designed so that utility-maximizing network participants pursuing selfish goals contribute to the network's overall security. Deviating from the agreed-on protocol would directly incur financial losses and is hence neither rational, nor sustainable. Due to the properties described above, blockchains are commonly considered to be trustworthy systems, i.e., transactions are always processed correctly and the results are reliable.

However, for many applications blockchain-internal data is not sufficient; they require data from blockchain-external systems like databases or sensors. In this case, the blockchain needs to rely on data provided by a third party which cannot be validated within the network. This cannot happen naively; the data needs to be trustworthy! Otherwise, trustworthiness of on-chain applications ends when external data becomes part of transactions.

Various so-called Oracle systems have been proposed, which aim to provide data to a blockchain in a more reliable way by incorporating techniques such as advanced cryptography [1], trusted execution environments [2], and voting [3]–[6].

However, these systems greatly differ in assumptions and applicability. They only give attention to specific aspects of the larger challenge of bringing data to the blockchain in a trustworthy way. Since the approaches are very different from each other and vary in their - often narrow - scope and applicability, they are hard to compare.

To address this shortcoming and provide a conclusive framework, in this paper, we explore and define key properties of *Trustworthy Data On-chaining Systems*. In this context, we make two main contributions:

1) We derive requirements for Trustworthy Data On-chaining Systems, hereby taking as a holistic view on the process of bringing data from the blochchain-external world onto the blockchain. We argue that the established distributed systems properties of safety and liveness are not sufficient and propose *truthfulness* as a third, fundamentally critical requirement to adequately characterize Trustworthy Data On-chaining Systems.
2) We provide an in-depth assessment and comparison of state-of-the art Oracle approaches with regards to the criteria for Trustworthy Data On-chaining Systems. This allows us to identify weaknesses of current approaches and deduct possibilities for future research.

This paper is organized as follows: In Section II, we describe the narrow perspective of current Oracle systems and motivate the need for a more holistic view. We respond to this problem by introducing Data On-chaining as a comprehensive approach for provisioning blockchain-external data to smart contracts in Section III. In Section IV, we derive requirements for Trustworthy Data On-Chaining Systems and identify truthfulness as a fundamentally new requirement. Taking these results into account, we proceed to categorize and compare existing oracle implementations, before we discuss their limitations and applicability in Sections V and VI.

## II. Problem Statement

Due to the security guarantees provided by blockchains, smart contracts are often used to implement critical application logic. The contracts may require external data as input for executing on-chain state transitions. However, the involvement

978-1-7281-4693-5/19/$31.00 ©2019 IEEE
DOI 10.1109/Blockchain.2019.00075

of external data for smart contract computations implies a risk: data can be unreliable, i.e., unavailable, maliciously modified, or untruthfully reported. Unreliable data used for smart contract computations may trigger blockchain transactions that should not be triggered and, thereby, cause economical damage, i.e., financial loss. As a consequence, the security guarantees of the smart contract are devalued and the trust and the reliability that blockchains are applied for are threatened.

The reliable provisioning of blockchain-external data to smart contracts has been addressed by a variety of Oracle proposals. *Oracles* are middleware systems situated at the edge of the blockchain that intermediate between smart contracts and external systems. TownCrier [2], for example, is an Oracle that employs Trusted Execution Environments (TEE) to guarantee tamper-proof data provisioning for smart contracts. A different approach is Astraea [3] that introduces game-theoretical mechanisms to incentivize truthful provisioning of data. Further, TLS-N [1] proposes a modification to the Transport Layer Security (TLS) protocol that allows to prove non-repudiation of data obtained in TLS sessions to third parties. What those Oracle proposals all have in common, is the intent to make off-chain data available to smart contracts in a reliable manner. These Oracle approaches, however, differ fundamentally in their system assumptions, trust model, applicability, and in the specific qualities that they aim to address. For example, TownCrier [2] and TLS-N [1] address data authenticity and integrity, whereas Astraea motivates truthful data provisioning. Further, those approaches take a limited perspective on the problem of providing unreliable data by focussing only on the Oracle system component itself and addressing only partial aspects of the larger challenge of assuring trustworthiness in Data On-chaining Systems.

The diversity of existing Oracle approaches makes them hard to compare and to reason about their qualities. To date, no systematic description of Oracles has been provided that allows this comparison and supports the design of novel approaches.

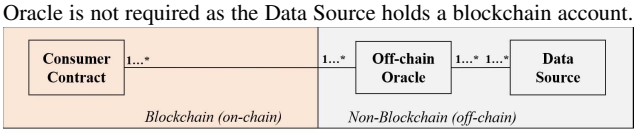
## III. Data On-chaining

Data On-chaining is the first concept that approaches the reliable provisioning of data for smart contracts from a holistic system perspective. It, hence, allows for comparability of existing Oracle approaches and paves the way for future research on Data On-chaining Systems. By addressing the problem of having unreliable data infiltrate smart contracts and, hence, devalue the security guarantees provided by the blockchain, we define *Data On-chaining* as the reliable provisioning of blockchain-external data to smart contracts in Data On-chaining Systems which consist of on-chain and off-chain components.

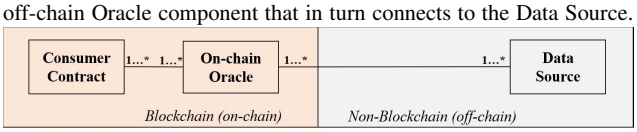
In contrast to the term *Oracle*, which takes a narrow view on the middleware component, Data On-chaining takes an integrated system perspective comprising all three components of a Data On-chaining System: the external Data Source, the Oracle, and Consumer Contract as described in Section III-A. This extended system perspective helps to raise awareness for aspects beyond those concerning Oracle exclusively, such as the availability and truthfulness of Data Sources.

Further, on-chaining complements the term *off-chaining* which describes the outsourcing of storage and computation to non-blockchain systems while preserving the blockchain's safety and liveness [7]–[10]. On-chaining takes the opposite direction and caters for the integration of external resources. Data On-chaining particularly focusses on providing data.

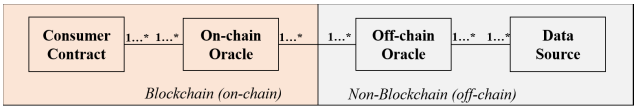

(a) Consumer Contract and Data Source can connect directly. An Oracle is not required as the Data Source holds a blockchain account.




(b) The Consumer Contract connects to the blockchain account of the off-chain Oracle component that in turn connects to the Data Source.




(c) The Consumer Contract connects to an on-chain Oracle component that in turn connects to the Data Source's blockchain account.




(d) The Consumer Contract connects to an on-chain Oracle component that communicates with the blockchain account held by the off-chain Oracle component which in turn connects to the Data Source.

Fig. 1: Data On-chaining System Architectures

### A. System Model

Data On-chaining Systems to consist of at least two components, the on-chain Consumer Contract and the external Data Source, and one optional component, the Oracle. Figure 1 summarizes the different architectural compositions of those components. Implementations of the system architectures depicted in Figures 1b, 1c, and 1d are described in Section V.

The *Data Source* is an off-chain system that is in possession of the information required by the Consumer Contract. Data Sources can be of different types, including web-based news feeds or stock brokers, sensors and devices in the Internet of Things, but also humans that provide their knowledge by means of blockchain transactions or web-technologies. Consumer Contracts may obtain data from multiple Data Sources directly or indirectly. If a Data Source maintains a blockchain account, it is able to communicate directly with Consumer Contracts as shown in Figure 1a. Otherwise an Oracle is required for intermediation (compare Figures 1b, 1c, 1d).

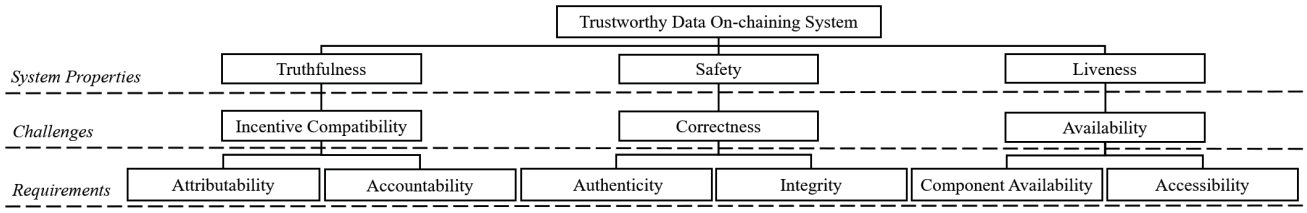



Fig. 2: Requirements for Trustworthy Data On-chaining Systems

The *Oracle*, as we understand it in Data On-chaining Systems, is an intermediate system component that bridges communication between the on-chain Consumer Contract and the off-chain Data Source. Oracles are not needed for Data On-chaining if the Data Source maintains a blockchain account as depicted in Figure 1a. The composition of an Oracle depends on the capabilities and requirements of Consumer Contracts and Data Sources: If the Data Source does not maintain a blockchain account, an off-chain Oracle component is required as depicted in Figure 1b and 1d. As depicted in Figure 1c and 1d, an on-chain Oracle component is required in some cases for bridging communication from off-chain to on-chain. Depending on the system requirements Consumer Contracts may employ multiple Oracle and Oracles, in turn, may consult multiple Data Sources.

The application logic in Data On-chaining Systems is implemented in one or multiple *Consumer Contracts*. Consumer Contracts require external data as input for on-chain state transitions. They can only be addressed via blockchain transactions and, thus, they can only receive messages from smart contracts or blockchain accounts maintained by external entities. Examples for applications of Consumer Contracts are blockchain-based insurance policies [11]–[13], purchase management in marketplaces [14]–[17] or goods tracking in supply chain networks [18], [19]. In the remainder of this paper, we assume a single Consumer Contract.

As the on-chain component of Oracles and the Consumer Contract are smart contracts, they are trustworthy, that is, they are highly available, tamper-proof, and truthful. Being situated outside the blockchain, the off-chain component of the Oracle and the Data Source, however, are not trustworthy by default but vulnerable for failures and attacks.

### B. Process Model

In on-chaining systems, data provisioning can be push- and pull-based. In a *pull-based* model, the Consumer Contract explicitly requests data from the Data Source, whereas in a *push-based* model, the Data Source provides data without previous requests. The Oracle forwards data from the Data Source to the Consumer Contract and, as required in the pull-based model, the request from Consumer Contract to the Data Source. This intermediation may involve data and request transformations to meet the format requirements of Data Sources and Consumer Contracts. Regardless the initiation of data provisioning, data has two states while being in the Data On-chaining System. Data is *in transit* during transmission, that is, it is sent from one component to another component. Data is *at rest* while it is not in transmission but controlled by one component. Data becomes part of the on-chaining system once it is received or recorded by the Data Source. Earlier states are not considered.

## IV. Requirements for Trustworthy Data On-chaining Systems

*Safety* and *liveness* are established properties to characterize distributed systems for more than 40 years [20]. In the context of Data On-chaining Systems, safety means that no state transition of the blockchain is triggered by incorrect data, whereas liveness means that blockchain state transitions are never not executed due to unavailable data. However, the properties safety and liveness do not fully capture the notion of trustworthiness in blockchain-based systems. As a third key property for Trustworthy Data On-chaining Systems, we introduce *truthfulness*. Truthfulness means that no execution of blockchain state transition is caused by untruthful data provisioning, but instead, data is always provisioned in a well intended way and to the best of the Data Source's knowledge.

From those three properties for Trustworthy Data On-chaining Systems, we derive the challenges *incentive compatibility*, *correctness*, and *availability* and introduce associated requirements that mirror the challenges in more detail. An overview of the system properties, challenges, and requirements is provided in Figure 2.

### A. Incentive Compatibility

The novel challenge that derives from truthfulness is incentive compatibility. Incentive compatibility is a system property that holds if system components are *accountable* and *attributable* for their actions, i.e., they have something at stake and the data provisioned as well as changes to the data are attributable to the originator.

In incentive compatible systems, it is assumed that components are *homini oeconomici* whose primary goal is to maximize their individual utility. If this economic rationality holds, financial incentives can be placed in a way that motivates truthful reporting and intermediation of data. Such incentives are realized by monetary rewards and penalties that respectively apply for truthful and untruthful data provisioning.

*1) Attributability:* The data provisioned to the Consumer Contract must be attributable to the responsible Data Source and modifications to the data during intermediation must be attributable to the Oracle.

To apply incentives to Data Sources, it must be possible to map provisioned data to its provider and, hence, to identify the Data Source that provisioned data untruthfully. Based on such an attribution of behavior, Data Sources can be rewarded and penalized for their actions. The same holds for any data modifications caused by the Oracle.

*2) Accountability:* Components in Data On-chaining Systems must have something at stake that makes them accountable for truthful data provisioning.

To realize accountability, Data Sources, for example, make a deposit before providing data. This deposit is payed back if truthful data provisioning is observed by the system, otherwise it is slashed. The motivation for Data Sources to provide data at all, is a reward that is payed out for truthful data provisioning. The rewards can be enriched with deposits that are withheld by the system as penalty.

### B. Correctness

The challenge that derives from system safety is correctness. Correctness is defined as a property of data that holds if the data is *authentic* and *of integrity*. The Consumer Contract must be able to verify that the data obtained really originates from the specified Data Source and that it has neither maliciously been modified nor unintentionally been changed. Approaches to data correctness are discussed in Section V-A, and V-B.

*1) Authenticity:* The Consumer Contract must be able to verify the authenticity of data received from the Data Source.

Data authenticity is required to prevent attackers from triggering unauthorized blockchain transactions by pretending to be the actual Data Source. As established security problem, authenticity can be implemented by means of asymmetric cryptography, such as digital signatures. However, the verification of the Data Source's identity is challenging due to the limited communication capabilities of the smart contracts. For example, the Consumer Contract cannot simply access Public Key Infrastructure (PKI) to approve a web-based Data Source's identity in form of its public key, but instead requires support of a trusted intermediary.

*2) Integrity:* The Consumer Contract must be able to verify that the data received has not maliciously been tampered with and has not unintentionally been changed.

Preserving data integrity is important for two reasons. First, it prevents malicious manipulations of attackers and, second, it prevents the data from being unintentionally modified through defectiveness, e.g. caused by malfunctioning communication channels. Both possibilities have equally drastic consequences on the safety of the on-chaining system, as critical Consumer Contract computation may be erroneously triggered. Integrity needs to be preserved in all cases, that are, data in transit and data at rest. It can be implemented by means of cryptographic techniques.

### C. Availability

The challenge that derives from system liveness is availability. Availability is defined as a property of data that holds if it can be accessed whenever access is required. It is achieved if all components of the on-chaining application are *available* and the data maintained by the components are *accessible*.

*1) Component Availability:* The off-chain system components, Oracle and Data Source, must be designed in a way that minimizes outages.

As data is replicated on every node in globally distributed blockchain networks like in Ethereum [21], the blockchain provides high availability guarantees compared to most non-blockchain system. The availability of Data On-chaining Systems, however, is as good as the availability of its least available component, which is either the off-chain Oracle component or the Data Source. To compromise the availability of on-chaining applications as little as possible, it is, hence, important to minimize the delta between the availability of the off-chain components and the one of the Consumer Contract. As a consequence, the off-chain components must be replicated.

*2) Accessibility:* Data controlled by the Data Source or by the Oracle must be accessible in the sense that it can be provisioned, unrestrictedly and at any time.

The availability of system components does not automatically guarantee availability of the maintained data as the access to the data can be restricted. In public blockchains, transaction fees, required to pay the blockchain miners, are carried by the party that initiates the transactions. In Ethereum, for example, the costs of transactions are determined by their complexity and accounted for in an Ethereum-specific unit, called *Gas*, which can be purchased for Ether. If smart contracts or external blockchain account holders in Ethereum-based on-chaining applications don't have enough Ether, they cannot submit transactions and, hence, cannot provide the required data. Thus, to guarantee accessibility of data in public blockchain-based Data On-chaining Systems, providing components with sufficient funds to pay transaction fees is indispensible.

## V. Oracle Approaches

In this section, we now can discuss and differentiate approaches to Oracles using the requirements that we derived from the Data On-chaining challenges and system properties.

### A. TLS-based Data On-chaining

In TLS-based Data On-chaining, the Transport Layer Security (TLS) protocol is instrumented to assure authenticity and integrity of the data provisioning by the Data Source.

The TLS protocol guarantees message integrity and server authentication through the exchange of certificates during the TLS-handshake. The certificate allows clients to validate the server's identity by consulting Certificate Authorities that are part of a Public Key Infrastructure (PKI). As de-facto standard for secure communication in the internet, TLS is supported by most web servers, involving millions of potential Data Sources that are registered in a PKI.

*1) Components:* The system consists of the Consumer Contract, an HTTPs-enabled Data Source, and an off-chain Oracle component (OffOC) as depicted in Figure 3. The Data Source does not maintain a blockchain account and, thus, an

OffOC is required for intermediation. The data provisioning model is push-based.

*2) Concept:* The OffOC and the TLS-enabled server establish a TLS connection. The integrity and authenticity of data obtained from the server during this TLS session can be verified by the OffOC after validating the server's certificate with the PKI. If the OffOC is trusted, it signs the data, transforms it into a blockchain transaction and submits it to the Consumer Contract as blockchain transaction. The Consumer Contract then verifies the OffOC's signature. If the OffOC is not trusted, it first constructs a proof from the TLS-session that allows the Consumer Contract to verify integrity and authenticity of the obtained data with the Data Source's certificate. Proof and data are then submitted as blockchain transaction to the Consumer Contract who verifies authenticity and integrity using the Data Source's certificate.

The challenges for TLS-based Data On-chaining are (1) to construct such a proof and (2) to allow the Consumer Contract to verify the Data Source's certificate.

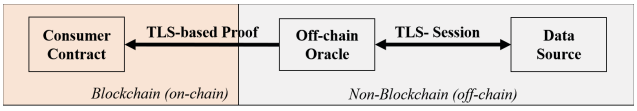



Fig. 3: TLS-based Data On-chaining

*3) Implementations: TLS-Notary* [22] allows a client to prove the authenticity and integrity of data obtained from a server during a TLS session to a specified third party, the auditor. The auditor must simultaneously be connected to the client. As currently no smart contract implementation of such an auditor exist, the auditor cannot be represented by the Consumer Contract itself, but instead, is represented by a trusted OffOC. For example, Oraclize [23] applies TLS-Notary by implementing the auditor on an Amazon EC2 instance and thereby exploits the reputation of Amazon Web Services as trusted host. TLS-Notary works as follows: During the TLS handshake between client and server, the auditor provides the client with the pre-master secret data required to compute the master secret, but withholds the part that is needed to generate the MAC key. Without the MAC key the client cannot decrypt and authenticate the traffic received from the server. After the client received encrypted responses to its requests, it generates a hash from the exchanged traffic and sends it to the auditor as a commitment to the session. In return, the client receives the missing part of the pre-master secret from the auditor which enables decryption and authentication of the traffic. The TLS-Notary proof consists of the pre-master secret and the hashed TLS session provided by the client. The pre-master secret allows the auditor to verify authenticity of the TLS session. Withholding the MAC key hinders the client to fake the requested content of the TLS session because it can neither decrypt the content received, nor can it encrypt a fake session as the auditor can check the TLS handshake.

*TLS-N* [1] is a modification to the TLS protocol that overcomes the requirement of the client in TLS-Notary to connect to the auditor by generating a non-repudiation proof that can independently be verified by any third party that has access to the PKI. This removes the trust assumption from the OffOC. However, it proposes a modification to the TLS protocol that is not supported by most public web servers. Further, the Consumer Contract cannot directly approach the PKI to obtain the server's public key, but instead must therefore rely on a trusted intermediation. In the TLS-N protocol, the server maintains a state of the TLS session established with the client. The state consists of a continuously updated hash value of the exchanged traffic, an ordering vector, and a start timestamp. Once the client requests the proof, the server signs the state using its public key and sends it to the client. The proof is constructed from the session data maintained by the client and the state that is signed by the server. The client also integrates the server's certificate into the proof as well as all ancestor certificates. This allows any third party that has access to the PKI to verify the server's identity by checking its public key.

### *B. Enclave-based Data On-chaining*

Similar to TLS-based Data On-chaining, Enclave-based Data On-chaining instruments TLS to guarantee data authenticity and integrity. However, to remove the trust assumption from the Oracle, *Trusted Execution Environments* (TEE) are applied that attest data authenticity and integrity to the Consumer Contract.

TEEs provide an execution environment for computations that is confidential and preserves integrity. Technically, TEEs are realized as a set of instructions extending the functionality of a processor with hardware enforced security policies that protect a dedicated address space against unauthorized access. To verify the identity of an application that runs inside an enclave, external parties can consult a Remote Attestation Service (RAS) that is usually maintained by the manufacturer of the TEE, e.g. Intel for Intel SGX [24]. Similar to PKIs, the RAS attests the validity of the enclave's public key that is used for message authentication.

*1) Components:* The system consists of the Consumer Contract, an HTTPs-enabled Data Source, and an off-chain Oracle component (OffOC). The Data Source does not hold a blockchain account and, thus, an OffOC is required for intermediation. The host that implements the OffOC, however, runs a TEE that allows to protect critical computation and data against malicious modifications. The data provisioning model can be pull- or push-based.

*2) Concept:* The OffOC establishes a TLS connection to the Data Source. As first step, integrity and authenticity of the data obtained in this TLS-session are verified by the OffOC inside the TEE using TLS capabilities, namely the server's certificate and the PKI. As second step, the OffOC signs the data using the private key that is securely maintained inside the TEE and forwards the signed data to the Consumer Contract. The Consumer Contract then verifies the signature using the public key of the TEE that it reviews in advance by consulting the RAS. In Enclave-based Data On-chaining, authenticity and integrity of the Data Source's messages are

proven in a two-step validation chain, involving TLS-based verification between the OffOC and the HTTPs-enabled Data Source as first step, and a digital signature between OffOC and Consumer Contract as second step.

The challenges for Enclave-based Data On-chaining are to allow the Consumer Contract to verify (1) the Data Source's certificate and (2) the enclave's public key.

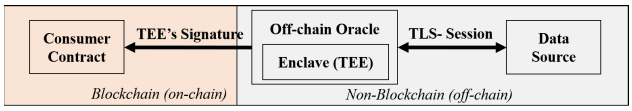



Fig. 4: Enclave-based Data On-chaining

*3) Implementation:* TownCrier [2] is an Enclave-based Oracle that enables pull-based data provisioning and employs an additional on-chain Oracle that takes requests from the Consumer Contract and verifies signed data on behalf of the Consumer Contract. The TEE is implemented using Intel Software Guard Extensions (SGX) [24] that are built into specific Intel CPUs and rely on the Intel Attestation Service to verify the TEE's identity. As Intel SGX lacks direct network access, some parts of the OffOC run outside the enclave managing low-level network access. However, critical functionality including TLS handshakes and all cryptographic operations are executed inside the enclave to ensure security. To communicate with the smart contract, the TEE-based OffOC implements a blockchain client. Due to the requirement of participating in the consensus protocol, the blockchain client is implemented outside the enclave but provides an interface to the enclave to read and write transactions. All messages sent from inside the enclave are signed by the TEE's private key to ensure authenticity and integrity. By using the Intel SGX to secure critical parts of the TLS-protocol, TownCrier removes the trust assumption from the OffOC. The Consumer Contract, however, requires access to the Intel's Attestation Service (IAS) to validate the enclave's identity. This requires trust in the off-chain IAS infrastructures, which is consulted for identity verification.

### C. Voting-based Data On-chaining

Voting-based Data On-chaining motivates truthful data provisioning of Data Sources that exhibit economically rational behavior and are incentivized by monetary rewards and penalties to truthfully participate in votes.

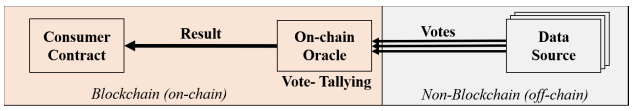



Fig. 5: Voting-based Data On-chaining

*1) Components:* The system consists of the Consumer Contract, multiple Data Sources, and an on-chain Oracle component (OnOC) as depicted in Figure 5. The Data Sources maintain a blockchain account and submit blockchains transactions to the OnOC. Consumer Contract and OnOC communicate on-chain. The data provisioning model is pull-based.

*2) Concept:* The Consumer Contract submits a data request to the OnOC in form of a question and thereby initiates a vote. The OnOC implements all voting logic. The request is attached with a *reward* that is paid to the Data Sources that vote for the right answer. This reward motivates Data Sources to participate and to truthfully provide data. The right answer is determined as the one that received most votes. However, to avoid that all answers are different from each other and instead facilitate the formation of a majority, the quantity of possible answers is restricted. For example, the question can only be answered with yes and no, or with integer values in the range from 0 to 9. After a specified time-window closes, the votes are tallied by the OnOC, and the right answer is determined. The provided reward is split and the shares are paid to the Data Sources that voted for the right answer.

To strengthen the commitment that Data Sources have to their vote, a *deposit* can be demanded from Data Sources for participation. Deposits are only paid back to the Data Sources that provide the right answer, otherwise, they are retained and increase the rewards payable in future voting rounds. Deposits can also be used for *weighted votes*. In weighted votes, the level of the deposit determines the weight that a Data Source has in the vote and the level of the reward that is emitted if the right answer is provided. The higher the deposit, the heavier the weight and the higher rewards and penalties.

To reward continuous truthful participation in votes, the behavior of the Data Sources can be tracked over the course of multiple voting rounds and assessed by a *reputation* value maintained by the OnOC. The reputation improves if a Data Source votes for the right answer, whereas it deteriorates if a false answer is provided. Like deposits, the level of the reputation value can impact the weight that Data Sources have in votes and the level of the reward that can be won.

The challenges in Voting-based Data On-chaining are (1) the right setting of all system parameters including rewards and deposits and (2) the protection against *Sybil* [27] and *lazy voting* attacks. Lazy voting attacks [3], [4] describe a situation where voters always vote for the same outcome refusing truthful data provisioning for the sake of laziness.

*3) Implementations:* Astraea [3] and Shintaku [4] are Voting-based Oracle proposals that construct a Nash Equilibrium within a set of game-theoretical rules. In both proposals, true, false, and unknown are the only possible answers.

Astraea [3] proposes a two-part voting scheme. In addition to the Consumer Contract, called proposer, that publishes data requests, called propositions, and bounties that are paid for voting correctly, two types of participants exist: the voters and the certifiers. Voters and certifiers vote independently from each other. While voters pay a relatively small deposit to vote for a proposal to which they are randomly assigned, certifiers pay a relatively high deposit to vote for a proposition of their choice. Rewards and penalties are paid according to the following structure: If voters and certifiers agree, participants of both types are rewarded if they agree with the majority and

TABLE I: Comparison of Oracle Approaches

| Approach | Implementation | *Attributability* | *Accountability* | *Authenticity* | *Integrity* | *Component Availability* | *Accessibility* |
|---|---|---|---|---|---|---|---|
| TLS-based Data On-chaining | TLS-Notary [22] | Data Source is attributable | Components have nothing at stake | Guaranteed through TLS & PKI | Guaranteed through TLS & PKI | OffOC is a single point of failure; data source availability is use case dependant | TX carried by OffOC; OffOC may withhold data |
| | TLS-N [1] | Data Source is attributable | Components have nothing at stake | Guaranteed through TLS & PKI | Guaranteed through TLS & PKI | OffOC is a single point of failure; data source availability is use case dependant | TX carried by OffOC; OffOC may withhold data |
| Enclave-based Data On-chaining | TownCrier [2] | Data Source is attributable | Components have nothing at stake | Guaranteed through TLS, PKI, enclave's signature & Remote Attestation Service | Guaranteed through TLS, PKI, enclave's signature & Remote Attestation Service | Enclave is a single point of failure & Data Source availabity is use case dependant | TX costs explicitly reimbursed for critical components |
| Voting-based Data On-chaining | Astraea [3] | Data sources are attributable per voting round | Data Sources are accountable per round through deposits | Verifiable on the blockchain | Guaranteed through blockchain transactions | Availability depends on number of participating Data Sources | Data provisioning incentivized & TX costs carried by Data Source |
| | Shintaku [4] | Data sources are attributable per voting round | Data Source are accountable per round through deposits | Verifiable on the blockchain | Guaranteed through blockchain transactions | Availability depends on number of participating Data Sources | Data provisioning incentivized & TX costs carried by Data Source |
| | Hivemind [25], Augur [6] | Data Sources are attributable across multiple voting rounds | Data Sources are accountable across multiple rounds through financial stakes in the system | Verifiable on the blockchain | Guaranteed through blockchain transactions | Availability depends on number of participating Data Sources | Data provisioning incentivized & TX costs carried by Data Source |
| | Chainlink [26] | Data Sources are attributable across multiple voting rounds | Data Sources are accountable across multiple rounds through financial stakes and reputation in the system | Verifiable on the blockchain | Guaranteed through blockchain transactions | Availability depends on number of participating Data Sources | Data provisioning incentivized & TX costs carried by Data Source |

penalized if not. Two reward pools exist for certifiers, one for true and one for false. Rewards for voters are paid from the proposition bounty and rewards for certifiers are paid from the certifiers reward pools. Penalties of both, voters and certifiers, are funded into the certifiers reward pool of the opposite value (penalties for incorrect *true* votes are assigned to the certifiers reward pool for *false* votes and vice versa). If voters and certifiers disagree, certifiers lose all their deposits (paid in both reward pools equally) but voters are neither rewarded nor penalized. In this case, the proposer loses its bounty which is paid equally to both certifier reward pools.

Shintaku [4] proposes a double-voting scheme. Different from Astraea, in Shintaku all voters are equal. However, voters pay a deposit which economically commits them to voting for two propositions to which they are both randomly assigned in the same voting round. As additional rule, voters may only receive rewards and penalties if they vote for both propositions with opposite values (true and false or false and true). If they do so, voters are independently rewarded and penalized for each of both assigned votes. If voters provide the same answer to both propositions, they are neither rewarded nor penalized and, hence, receive their deposit.

Hivemind [25] and Augur [6] are prediction markets that apply weighted voting and offer their own tokens for sale to let participants obtain stake in the system. The amount of tokens declares how much weight participants have in votes. In those approaches single nodes report real world events to decide prediction markets by staking reputation tokens. The more tokens are at stake, the more difficult it becomes to challenge the data reported for the market. Only by staking more tokens than the initial reporters, others can challenge the report and enter a dispute phase.

Reputation-based votes are realized in ChainLink [26]. ChainLink allows Consumer Contracts acting as clients to rate the Data Sources based on multiple factors, including the number of assigned, accepted and completed requests, the average time to respond and the amount of penalty payments. ChainLink maintains those reputation ratings and considers them in the assessment of obtained responses. Higher reputation values allow for higher rewards and heavier weight in votes.

A comparison of the approaches and implementations with regards to the identified requirements for Trustworthy Data On-chaining is provided in Table I.

## VI. Discussion and Applicability

*TLS-based Data On-chaining* focusses on data correctness and thereby assures authenticity and integrity. The remaining requirements, however, are neglected. This implies that data sources can lie and Oracles can withhold data without being accounted. Also, the availability guarantees are low as the Off-chain Oracle Component (OffOC) represents a single point of failure. However, with regards to data correctness, TLS-based Data On-chaining is especially powerful if a proof can be constructed from the unmodified TLS protocol that attests authenticity and integrity to the Consumer Contract. In this case, the OffOC is not required to be trusted and millions of Data Sources can securely be accessed.

*Enclave-based Data On-chaining* addresses data correctness and employs Trusted Execution Environments (TEE) to remove the need to trust the OffOC. TEEs enable secure off-chain computation capacities that could, for example, be used to transform malformed data into the format required by the Consumer Contract. However, the security assumption

for TEEs is strong and TEEs have been proven vulnerable [28]–[30]. Also, Remote Attestation Services [24] required to authenticate the TEE must be trusted.

*Voting-based Data On-chaining* explicitly addresses incentive compatibility by integrating economic rewards and penalties in the system design. Data availability is high if the number of Data Sources is high which in turn depends on the parametrization of the incentive system. As transactions accepted by the blockchain are transparent and tamper-resistant, the authenticity of transactions submitted by the Data Sources can be verified and their integrity is guaranteed. With regards to the requirements for Trustworthy Data On-chaining, Voting-based Data On-chaining can be evaluated as highly trustworthy. However, its applicability is strongly restricted as Data Sources can only confirm information that is publicly accessible. In cases where specialized data is required that is held by a single device, Voting-based Data On-chaining is not applicable.

## VII. Conclusion

In this paper, we introduced Trustworthy Data On-chaining, a novel holistic perspective on reliable data provisioning for smart contracts that allows for evaluating existing Oracle approaches and provides a framework for future designs of Trustworthy Data On-chaining Systems. Further, we provided a profound analysis and a detailed comparison of existing Oracle approaches with regards to requirements that we derived from challenges for Trustworthy Data On-chaining.

As the notion of trustworthiness in blockchain-based systems is not adequately addressed by safety and liveness, we presented *truthfulness* as novel property required to characterize Trustworthy Data On-chaining Systems. Truthfulness is not only important in the context of blockchains, but a generic requirement for trustworthy systems, e.g., IoT systems that build on sensor data provided by a group of untrusted entities.

We conclude that future Data On-chaining Systems need to consider not only safety and liveness, but also truthfulness from the start and hope to encourage further research on the open issues current proposals face.

## References


[1] H. Ritzdorf, K. Wüst, A. Gervais, G. Felley, and S. Capkun, "TLS-N: non-repudiation over TLS enablign ubiquitous content signing," in *NDSS*. The Internet Society, 2018.

[2] 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.

[3] J. Adler, R. Berryhill, A. G. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A decentralized blockchain oracle," *CoRR*, 2018.

[4] R. Kamiya, "Shintaku: An end-to-end-decentralized general-purpose blockchain oracle system," Available at: https://gitlab.com/shintaku-group/paper/blob/master/shintaku.pdf, 2018, acccessed in February 2019.

[5] S. Ellis, A. Juels, and S. Nazarov, "Gnosis whitepaper," Available at: https://gnosis.pm/resources/default/pdf/gnosis-whitepaper-DEC2017.pdf, 2017, acccessed in February 2019.

[6] J. Peterson, J. Krug, M. Zoltu, A. K. Williams, and S. Alexander, "Augur: a decentralized oracle and prediction market platform," Available at: http://bitcoinhivemind.com/papers/truthcoin-whitepaper.pdf, 2015, accessed in February 2019.

[7] J.Eberhardt and S.Tai, "On or Off the Blockchain? Insights on Off-Chaining Computation and Data," in *ESOCC 2017: 6th European Conference on Service-Oriented and Cloud Computing*, 2017.

[8] J. Eberhardt and J. Heiss, "Off-chaining models and approaches to off-chain computations," in *Proceedings of the 2Nd Workshop on Scalable and Resilient Infrastructures for Distributed Ledgers*, ser. SERIAL'18. ACM, 2018.

[9] G. Zyskind, O. Nathan, and A. Pentland, "Enigma: Decentralized computation platform with guaranteed privacy," *CoRR*, 2015.

[10] A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 2016, pp. 839–858.

[11] V. Gatteschi, F. Lamberti, C. Demartini, C. Pranteda, and V. Santamaría, "Blockchain and smart contracts for insurance: Is the technology mature enough?" *Future Internet*, vol. 10, 2018.

[12] H. T. Vo, L. Mehedy, M. Mohania, and E. Abebe, "Blockchain-based data management and analytics for micro-insurance applications," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, ser. CIKM '17. ACM, 2017.

[13] "Etherisc," Available at: https://etherisc.com/, accessed in February 2019.

[14] G. Fedak, W. Bendella, and E. Alves, "iexec: Blockchain-based decentralized cloud computing," Available at: https://iex.ec/wp-content/uploads/pdf/iExec-WPv3.0-English.pdf, 2018, acccessed in February 2019.

[15] Golem, "Golem project - crowdfunding whitepaper," Available at: https://golem.network/crowdfunding/Golemwhitepaper.pdf, 2016, accessed in February 2019.

[16] "Blogpv," Available at: http://www.blogpv.de, accessed in February 2019.

[17] M. Klems, J. Eberhardt, S. Tai, S. Härtlein, S. Buchholz, and A. Tidjani, "Trustless intermediation in blockchain-based decentralized service marketplaces," in *Service-Oriented Computing - 15th International Conference, ICSOC*, 2017.

[18] N. Kshetri, "Can blockchain strengthen the internet of things?" *IT Professional*, vol. 19, 2017.

[19] M. Westerkamp, F. Victor, and A. Küpper, "Blockchain-based supply chain traceability: Token recipes model manufacturing processes," *CoRR*, 2018.

[20] L. Lamport, "Proving the correctness of multiprocess programs," *IEEE Trans. Softw. Eng.*, 1977.

[21] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," *https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper*, 2014.

[22] TLSNotary, "Tlsnotary - a mechanism for independently audited https sessions," Available at: https://tlsnotary.org/TLSNotary.pdf, 2014, accessed in February 2019.

[23] Oraclize, "A scalable architecture for on-demand, untrusted delivery of entropy," Available at: www.oraclize.it, acccessed in February 2019.

[24] Intel, "Innovative Technology for CPU Based Attestation and Sealing," https://software.intel.com/en-us/articles/innovative-technology-for-cpu-based-attestation-and-sealing, 2013.

[25] P. Sztorc, "Truthcoin: Peer-to-peer oracle system and prediction marketplace," Available at: http://www.augur.net/whitepaper.pdf, 2018, accessed in February 2019.

[26] S. Ellis, A. Juels, and S. Nazarov, "Chainlink a decentralized oracle network," Available at: https://link.smartcontract.com/whitepaper, 2017, acccessed in February 2019.

[27] J. R. Douceur, "The sybil attack," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Springer Berlin Heidelberg, 2002.

[28] F. Brasser, U. Müller, A. Dmitrienko, K. Kostiainen, S. Capkun, and A. Sadeghi, "Software grand exposure: SGX cache attacks are practical," *CoRR*, 2017.

[29] M. Schwarz, S. Weiser, D. Gruss, C. Maurice, and S. Mangard, "Malware guard extension: Using SGX to conceal cache attacks," *CoRR*, vol. abs/1702.08719, 2017.

[30] A. Moghimi, G. Irazoqui, and T. Eisenbarth, "Cachezoom: How SGX amplifies the power of cache attacks," *CoRR*, vol. abs/1703.06986, 2017.