Verify My Origin: Livestock DNA Fingerprinting Through Blockchain Oracle

1st Amirmohammad Pasdar
School of Computing
Macquarie University
Sydney, Australia
amirmohammad.pasdar@hdr.mq.edu.au

2nd Young Choon Lee
School of Computing
Macquarie University
Sydney, Australia
young.lee@mq.edu.au

3th Paul Ryan
Aglive Pty Ltd
Geelong, Australia
paul@aglive.com

4rd Zhongli Dong
Aglive Pty Ltd
Sydney, Australia
andrew@aglive.com

***Abstract*—Blockchain, a type of distributed ledger technology, has revolutionized the digital economy such as cryptocurrencies and supply chain management with its transparency, immutability, and decentralization properties. In addition, smart contracts are introduced to the blockchain to provide programmability removing third parties for administration. Although promising, blockchains and smart contracts are closed technologies meaning they have no interaction with the external world where real-world data and events exist, i.e., off-chain data. It becomes more challenging when the off-chain data is unstorable onto the blockchain due to data volume and privately maintained by third parties for security and confidentiality. In this paper, we address the problem of enabling a private blockchain platform to access privately owned sensitive off-chain data (i.e., DNA fingerprinting). This off-chain data is used for traceability of products (i.e., products' origin) along the supply chain with a real-world livestock use case. To this end, we present a livestock blockchain oracle (LBO) to mitigate the accessibility issue through a trusted and convenient way of verifying purchasable products for livestock DNA fingerprinting verification. We have conducted an evaluation study with real-world livestock data provided by third-party service providers. Results based on the livestock product information and registered DNA service providers show that LBO is a reliable and responsive decentralized oracle blockchain for verification.**

***Index Terms*—Blockchain, Smart contracts, Blockchain oracles, DNA Fingerprinting**

I. Introduction

Blockchain is a distributive technology that has revolutionized the digital economy and decentralized finance (Defi) ecosystem. The use of blockchain can be seen in decentralized applications (DApps), including but not limited to food security [1]. A blockchain is a form of distributed ledger technology where transactions are permanently stored onto many nodes. Transactions are messages digitally signed by cryptographic techniques for storing parameters/results and are immutable records collected in the form of blocks chained together through the hash. These blocks are appended to the ledger by consensus algorithms such as proof of work (PoW) or proof of stake (PoS) [2]. In addition, smart contracts are known as event-driven and self-executable programs that are compiled into bytecode. They are executed on the blockchain by employing available resources that attract transaction fees.

Smart contracts and blockchain technology have connectability issues meaning they have no access to real-world information and events (off-chain data). This information can be publicly available data (e.g., stock market data and exchange rates) or privately owned sensitive information in high volumes such as DNA fingerprinting in the agriculture industry. Hence, they are closed technologies meaning they have no interaction with the external world; therefore, the usability of smart contracts is limited to the data on the blockchain (on-chain data). There should be a mechanism referred to as the "blockchain oracle" to bring the real-world data in the blockchain and broaden the scope of smart contract operation. Oracles (or *data feeds*) are third-party services in the form of smart contracts deployed on the blockchain that send and verify the external information to the smart contracts [3]. They may consult with single or multiple sources (decentralized oracles) for fetching the required information.

There have been several studies on blockchain oracles, e.g., [4]–[7], for transferring data to the blockchain, but most if not all of them did not examine it from the private *industry* perspective. Chainlink [4] is a general-purpose and token-based framework for building secure decentralized input and output oracles. Town Crier [5] is an enclave-based oracle that employs Intel software guard extension (SGX) technology for ensuring data integrity. Adler et al., [6] present a general-purpose decentralized oracle referred to as Astraea that deals with binary queries, and Al Breiki et al., [7] propose a decentralized access control for the internet of things (IoT) data that is assisted by blockchain and trusted oracles.

In this paper, we address the problem of enabling the private blockchain platform to access sensitive off-chain data with a case study for livestock DNA fingerprinting verification. DNA fingerprinting provides the traceability of livestock products (e.g., beef) at any point along the supply chain by comparing its DNA profile with an initial privately maintained sample. Third-party DNA services are already in use on farms for livestock, and DNA testing units fitted to smartphones can test and return a result in minutes without the need to send it out to the lab. The challenge is to connect the blockchain platform with these third-party DNA service providers holding privately

Amirmohammad Pasdar (amir@aglive.com) and Zhongli Dong (andrew@aglive.com) are with Aglive Lab, P.O. Box 196, Geelong VIC 3220, Australia.

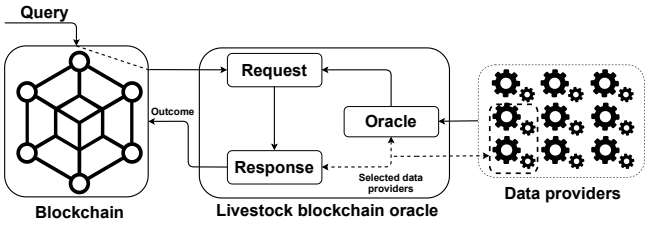


Fig. 1: Livestock blockchain oracle consists of three smart contracts; Oracle, Request, and Response.

owned sensitive data to provide a trusted and convenient way of verifying purchasable products.

To this end, we develop a livestock blockchain oracle (LBO shown in Figure 1) for DNA fingerprinting verification with real-world livestock data provided by third-party service providers. In particular, LBO employs three smart contracts: Oracle, Request, and Response. Oracle contract is in charge of registering the third-party APIs to be assigned to the queries submitted to the blockchain. The Request contract handles these queries, and it employs oracle reputation/performance metrics for selection to obtain the requested data. Returned results from the selected oracles are aggregated by the Response contract, where the performance and reputation metrics of participated data providers are adjusted.

We design and implement LBO that employs Web3j [8] as a lightweight Java and Android library for working with smart contracts and the Ethereum blockchain. We evaluate the performance of LBO based on real-world meat product information and a verified DNA service provider in terms of gas[1] consumption and turnaround time. Results present LBO as a reliable and responsive decentralized blockchain oracle.

The paper is organized as follows: Section II briefly overviews the related works on blockchain oracles and their design patterns. In Section III, we present the livestock blockchain oracle in detail.In Section IV, the oracle is evaluated and we conclude the paper in Section V.

II. Related Work

Blockchain oracles can be classified into three overlapping categories in terms of source, information direction, and trust. The source represents the origin of data, and the information direction implies how information flows (inbound or outbound) from/to external resources. Finally, trust can be centralized (single source) or decentralized (multiple sources).

In addition, from a technical perspective, blockchain oracles can be divided into two main groups based on how the outcome is finalized and saved onto the blockchain. They are referred to as voting-based oracles and reputation-based ones. Voting-based oracles, e.g., [6], [9]–[14] employ participants' stakes for outcome finalization while the reputation-based oracles, e.g., [4], [7], [15]–[18] consider reputation or performance metrics in conjunction with authenticity proof mechanisms for data correctness and integrity.

[1]Gas refers to the computational efforts required to execute specific operations on the Ethereum network.

Astraea [6] is a general-purpose decentralized oracle for binary queries. In this oracle, there are entities such as submitter, voter, and certifier, each holding stakes. Voters play low-risk/low-reward games while certifiers play high-risk/high-reward games, and based on their outcomes and whether they are matched, rewards/penalties are distributed. Kamiya [10] provides an extension to Astraea, and in the extended version, two propositions are submitted and based on the different responses, rewards are distributed. Also, Merlini et al., [9] present a paired-question oracle that employs two antithetic questions for queries. If responses to queries are matched, the submitter reclaims the bond, and voters are rewarded or penalized if they agree or disagree with the majority of answers, respectively. Nelaturu et al., [11] provide a framework that employs a crowd-sourced voting mechanism that uses strategies for oracles like Astraea [6].

Cai et al., [12] present a peer prediction-based protocol that employs a non-linear stake scaling with a lightweight scoring rule to control the rewards for the voters. Each report receives a score that rewards are distributed to the top-scored voters by considering the accuracy and degree of agreement with peers. Buterin [13] shows a mechanism that is based on the Schellingcoin concept for the creation of decentralized data feeds. In this mechanism, submitted responses are sorted, rewards are distributed to the users who provide correct responses, and it is between specific percentiles. Velocity [14] is a decentralized market for trading a custom type of derivative option and uses a smart contract called PriceGeth to fetch the price information in a real-time manner.

He et al., [16] present a scalable data feed service that uses a reputation evaluation strategy assisted by TLSNotary [15] to detect malicious nodes. Woo et al., [17] propose a distributed oracle to import time-variant data into the blockchain by employing multiple oracles, and data integrity is assessed with the help of Intel SGX. Al Breiki et al., [7] present a decentralized access control for IoT data that is assisted by trusted oracles. Wang et al., [18] propose an oracle based on Application-Specific Knowledge Engines (ASKE) framework to acquire and analyze information. It employs data analysis methods on the collected data managed by authoritative websites via web crawlers automatically. Last but not least, Chainlink [4] is a general-purpose and token-based framework that employs a blockchain-agnostic token that can simultaneously run on different blockchains through via external adapters for various data sources.

III. Livestock Blockchain Oracle (LBO)

This section describes the livestock blockchain oracle structure, and the problem is formulated. The symbols used in this paper are described in Table I.

A. The Blockchain Events

The Ethereum blockchain employs events and logs to facilitate the communication between smart contracts and decentralized applications as events are methods to understand a contract state changes. In this regard, smart contracts emit

TABLE I: Symbol description

Symbol	Description
$\mathbb{O}$	Data provider set
$\mathbb{R}$	Request set
$\mathbb{P}$	Response set
o_i	A data provider
ϕ_i	A data provider popularity
r_i/p_i	A request and response
$o_e/r_e/p_e$	Events of Oracle, Request, or Response
ω	Response time window
τ^r/τ_i^r	Response arrival time set and its time
ξ_p	Majority outcome of responses

events and record logs to the blockchain when a transaction is mined. In this case, the decentralized application (or the smart contracts) is notified to continue the process of a specific action. The livestock blockchain oracle uses the blockchain events (e) as a means of communication.

1) Oracle smart contract events: Events of the Oracle smart contract (o_e) can be grouped into two classes; registration events and controlling events. The former notifies the blockchain oracle, particularly the Request smart contract, that a new data provider is registered on the blockchain and can be employed to fetch requested query information. The latter refers to *controlling* events that restrict the registered data provider(s) in case of performance and trust issues. The controlling events trigger functions that activate/deactivate the registered data providers on the blockchain and inform the decentralized application.

Since the blockchain oracle relies on the performance metrics, the participated data providers may be banned to be involved in the process for a certain time due to intermittent issues, e.g., being unresponsive to the queries or being unqualified to retrieve the data. These data providers might be activated again after screening their performance.

2) Request smart contract events: Incoming requests to the blockchain oracle are managed by the Request smart contract, and this smart contract emits two main events (r_e). When a request is created, the corresponding event is emitted that mostly maintains the information about the product identifier and associated information.

Once a request is created, it should be assigned to registered and qualified data providers for fetching the information notifying the Response contract to take over the process. In addition, it includes information about selected data providers for the specific request.

3) Response smart contract events: When the Request events (p_e) are emitted, the Response contract will be in charge of fetching the required information by the involved data providers. It finalizes the feedback, returns the result to the blockchain, and updates performance metrics of the participated data providers. Thus, the Response smart contract emits events about when data providers return responses and whether the corresponding request becomes finalized.

Once data providers provide their responses, responses should be aggregated to return the final feedback to the blockchain and the application. The response contract will emit an event notifying the application that the corresponding request is finalized, attaching the final feedback and the corresponding data provider.

B. Smart Contracts

The overall structure of the designed blockchain oracle (Figure 1) presents how smart contracts interact with each other through emitted events on the blockchain. The designed oracle has three smart contracts; Oracle, Request, and Response. It is a form of a reputation-based oracle, meaning metrics of data providers are computed and assessed for selection. It employs *trusted* data providers for fetching data and submitting it to the blockchain.

1) Oracle: This smart contract is responsible for registering third-party data providers and maintaining their reputation metrics. These reputation metrics are kept to assist requests with selecting eligible data providers to obtain data.

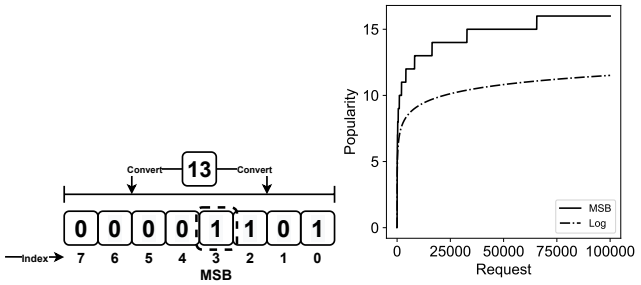
Data providers are initially registered in the system, given zero popularity value, and each data provider can be accessed by its address. Each registered data provider has a deterministic address on the blockchain whose popularity value is managed by a function that employs the most significant bit (MSB) procedure to increase/decrease the value with respect to the performance of data providers.

Figure 2 illustrates how MSB is calculated; in Figure 2a, the MSB is the largest index value in the binary format representation. Figure 2b shows the comparison between MSB and the natural logarithm, and the MSB mimics the logarithmic behavior in the real world as Solidity yet to support fixed-point number functionality. Hence, to control the popularity, MSB is employed that gradually and efficiently manages the value. This technique has two main benefits; firstly, the difference between popularity of data providers is not prone to sudden spikes and immediate changes. This feature enables the blockchain oracle to rely on the median value of their popularity fairly. Secondly, by considering Figure 2b, the sensitivity of updating the popularity is not linear or scaled to the number of served requests.

The Oracle contract also maintains the status of registered data providers and the monetary incentives/penalties in the form of balance that they may be given. Suppose the data provider is untruthful or has low performance. In that case, and regardless of their popularity value (recall the events of smart contracts; p_e and o_e), the data provider is deregistered and is not employed in the list of eligible data providers for fetching data.

2) Request: This smart contract maintains the requested query and holds information about assigned data providers with their provided stakes. This smart contract also relies on the Oracle contract to access the registered data providers.

When a request is created, it is given an identifier, including the arrival time. The remaining information will be filled while fetching the query data, such as the address of assigned data providers and their stakes. Once the request is finalized, the request status will be changed.



(a) MSB (b) Popularity

Fig. 2: MSB and popularity of data providers; (a) MSB definition and (b) comparison between MSB and the natural logarithm for behavior.

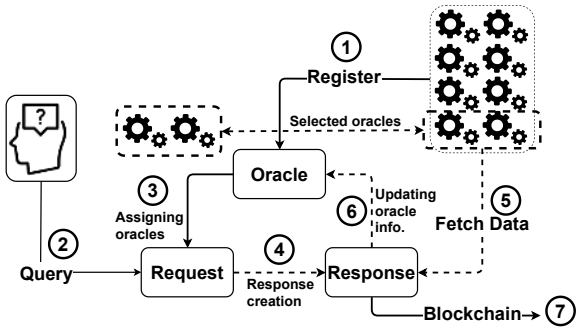


Fig. 3: Processing a request on the livestock blockchain oracle.

Upon creating a request, an event will trigger the blockchain function to assign registered data providers to the request. Data providers are assigned to the specific request based on their popularity values and registration status.

The Request contract employs the median value (m) technique for the selection of data providers within a set and has been used in the literature [13]. Per each request, the registration status of data providers is checked, the popularity values of the eligible data providers are sorted, and the median value is selected. This median value will act as a threshold for assigning data providers to the request to fetch the data. Hence, data providers with a larger popularity value than the median will be selected to retrieve data. As the median value is not affected by smaller or larger values, it becomes a suitable metric for choosing the data providers.

$$\mathbb{O}_i^r \leftarrow \{\forall\, o_i \in \mathbb{O}\ \exists\, m\ s.t.\ \phi_i \geq m\} \tag{1}$$

In Equation 1, m is the median value, and $\mathbb{O}_i^r$ is the list of eligible data providers to be assigned to the request (r_i). Once the list of data providers is ready, the Response contract will fetch the result for finalizing the outcome.

3) Response: The smart contract deals with fetching and storing responses of data providers and returning the final result (i.e., outcome) to the application.

Each response (p_i) maintains the block time, defined as a data provider's response arrival time. There is a response time window (ω) to ensure the data providers are responsive

Algorithm 1: Livestock blockchain oracle (LBO)

Data: Data providers ($\mathbb{O}$), request (r_i), responses ($\mathbb{P}_i$), response time window (ω)
Result: Outcome (γ_i^r)

Register new data providers, update $\mathbb{O}$, and emit o_e
Check the registration status of data providers in $\mathbb{O}$ and sort the list w.r.t. popularity in ascending order
Compute median value $m \leftarrow Median(\mathbb{O})$ and notify the Response contract by emitting r_e
Initialize $\mathbb{P}_i^r \leftarrow \emptyset$, $\tau_i^r \leftarrow \emptyset$ & $\mathbb{O}_i^r$ w.r.t. Equation 1
for $o_i \in \mathbb{O}$ **do**
  $\mathbb{P}_i^r \leftarrow p_i^o$ & $\tau_i^r \leftarrow Time(p_i^o)$
end
$\tau_p \leftarrow \min_{\forall \tau\ in\ \tau_i^r} \tau^r$
Compute $\xi_p \leftarrow \{\forall\, p_i \in \mathbb{P}\ s.t.\ |\tau_i^r - \tau_p| \leq \omega\}$
$\mathbb{P}_i^r \leftarrow \{\forall\, p_i \in \mathbb{P}\ \exists\ \tau_i^r\ s.t.\ |\tau_i^r - \tau_p| \leq \omega\ \&\ p_i = \xi_p\}$
$\gamma_i^r \leftarrow Median(\mathbb{P}_i^r)$ and emit p_e

in a reasonable time. Otherwise, they will be penalized (i.e., reducing their popularity and holding their stakes) and may be deactivated after repeatedly being unresponsive.

Selected data providers by the Request smart contract will be in charge of retrieving data for the Response contract. If responses are not received within a specified time window, they will be discarded, and their data providers will be penalized. In addition, data providers will also be penalized when their outcomes do not match with the majority. The majority rule is a decision for choosing the one that more than half of the outcomes support. Hence, for $|\mathbb{P}|$ outcomes received in the specified time window, there is ξ_p representing the outcome that more than half of data providers agree with. Equation 2 illustrates the increase ($\uparrow$) or decrease ($\downarrow$) of popularity (ϕ_i) and their corresponding stakes considering the least arrival time (τ_p) and the response arrival time of a data provider (τ_i^p).

$$\phi_i = \begin{cases} \uparrow & \forall\, p_i : |\tau_i^p - \tau_p| \leq \omega\ \&\ p_i = \xi_p \\ \downarrow & Otherwise. \end{cases} \tag{2}$$

Algorithm 1 in conjunction with Figure 3 illustrates how a request is processed by the livestock blockchain oracle. Figure 3 shows that per each incoming request, eligible data providers should be selected, and this eligibility is managed by Equation 1. Once the data providers are assigned to the request, the Response contract is notified by the event. The Response contract will maintain information about the retrieved data by data providers and their arrival times. Per a specified time window, the least arrival time of responses is detected and is used to filter non-responsive data providers. These data providers will be penalized by reducing their popularity and withholding their stakes based on Equation 2, and these updates will be saved onto the blockchain.

IV. Evaluation

In this section, we evaluate the livestock blockchain oracle performance. We first explain the simulation setup, and then

discuss the LBO performance result.

A. Simulation Setup

The smart contracts of the livestock blockchain oracle are written in the Solidity programming language (v0.6.0 and compatible with higher versions) and are converted into Java classes by Web3j library [8] to be compatible with a mobile application. Web3j is a lightweight Java and Android library for working with smart contracts and integrating with nodes on the Ethereum blockchain [19]. Solc-js compiles the smart contracts, and it is JavaScript bindings for the Solidity compiler [20]. Solc-js outputs the contract application binary interface (ABI) and a binary file (BIN) holding information about the hex-encoded binary for the transaction request.

Moreover, Ganache CLI is employed as a local Ethereum blockchain to test the decentralized application. It uses ethereumjs for simulating full client behavior, and remote procedure calls (RPC) functions [21].

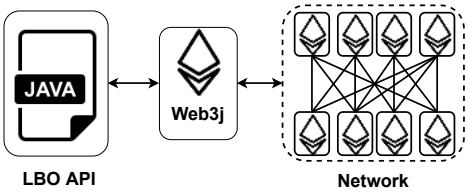


Fig. 4: Livestock blockchain oracle relies on Web3j library to communicate with the blockchain.

We use a sample real-world dataset that consists of essential livestock information including the product identifier code (PIC), RFID, and comprehensive analysis of its minerals.

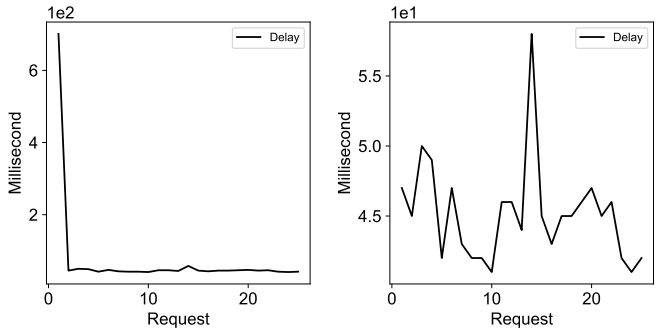
Also, a third-party service provider recognized in industry traceability is used as a trusted and registered data provider for the blockchain oracle. The dataset is privately owned and securely stored in their database due to the sensitiveness of the information. Hence, an API is provided for passing the product identifier for the dataset to output a binary result, i.e., valid or invalid. The API URL structure is shown in Equation 3 in which φ is the product RFID to be passed to the URL.

$$curl - X\ GET \quad \text{“}https://tracebaseapi.azurewebsites.net/RFID/\varphi\text{”} - H\ \text{“}accept : text/plain\text{”} \tag{3}$$

Randomly selected requests from the dataset are submitted to the blockchain for verification. We evaluate the performance of the designed blockchain oracle in terms of turnaround time and gas usage on the local Ethereum blockchain.

B. Simulation Results

We evaluate the livestock blockchain oracle based on a scenario in which the number of data providers is increased for scalability, and the consumed gas is reported. The consumed gas mainly shows that the designed blockchain oracle will not hit the existing gas limit for execution. Due to the limitation of the provided datasets, its sensitiveness and the way products are verified by the third party, the same provider is re-registered for each scenario. Since Request and Response contracts rely on the Oracle contract for interactions, the Oracle smart contract is deployed first, followed by the Request and Response smart contracts.



(a) The data provider API latency with very first request(s).

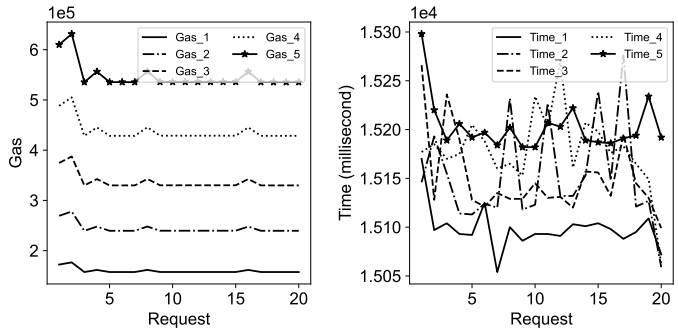
(b) The latency after the short initial spike period.

Fig. 5: Query response time of the third party API. The significant delay at the beginning in Figure 5a is due to the execution of underlying components, e.g., database query execution. Figure 5b can be seen as a microscopic view of Figure 5a after the short initial spiky delay.

We consider increasing the number of data providers to 5 to evaluate the scalability, and requests are randomly chosen from the dataset. The scalability evaluation will provide insights into how it may affect the performance of the designed oracle. Per each finalized request, the consumed gas on the Ethereum blockchain and the request turnaround time are reported. Since the responsiveness of the third-party API affects the designed blockchain oracle performance, the delay of queries is computed and shown in Figure 5.

Figure 5 presents how responsive the data provider is in terms of fetching the query data as the server load (i.e., the response time) affects the turnaround time of the livestock blockchain oracle. Hence, Figure 5a shows that there is a significant delay in returning the feedback for the initial request(s) submitted to the API due to execution of underlying components, e.g., database query execution, with the average ∼710ms. Apart from the initial delays (with a volatile average latency between ∼680ms and ∼730ms), the API has the average latency of ∼430ms with volatile response time, including spikes that is shown in Figure 5b. The Figure 5b points out that improving the response time explicitly affects the user experience achieved via optimizing the database and the webserver performance.

Figure 6 presents the livestock blockchain oracle performance in terms of consumed gas and the outcome's turnaround time. Figure 6a illustrates the consumed gas for all the scenarios where the number of data providers is increased. This Figure 6a demonstrates that by increasing the number of data providers for retrieving data, the gas consumption also



(a) Gas usage. (b) Outcome turnaround time.

Fig. 6: Livestock blockchain oracle performance, (a) gas usage, (b) the scalability evaluation and how long it takes to process a request. The number of participating data providers is shown as x in Gas_x (and $Time\_x$).

increases, and the gas consumption nearly has a linear relationship with the number of data providers in the system. The initial spike that is seen in the Figure could be related to the significant delay for requests (Figure 5a) and the underlying delays that exist between components of the blockchain oracle such as Web3j, Ganache CLI, and mining time.

Moreover, Figure 6b presents the scalability evaluation and shows how long the livestock blockchain oracle takes to finalize a request. The turnaround time for requests on average is $\sim$15s considering the underlying delay on the Ethereum blockchain for mining. In addition, the overhead communication time between the application and the blockchain should be considered. Increasing the number of data providers did not significantly impact finalizing the outcome and its turnaround time. The turnaround time seldom depends on the number of data providers, and it can be seen from Figure 6b, for the second half of the submitted requests (≥ 10), the turnaround time of 5 data providers is better than 2 or 4 data providers. In addition, the spikes in Figure 6b can be from the API response delay shown in Figure 5b as well as underlying performance issues with the Web3j and Ganache CLI [22]. In other respects, the designed oracle has the potential to be converted into a reliable and fast decentralized blockchain oracle.

V. Conclusion

Blockchain technology has revolutionized the digital economy and has changed the financial market in the last few years. This disruptive technology is a distributed ledger technology where data is shared among nodes connected over the internet. Data state changes on the blockchain are permanently saved onto the immutable ledger in a decentralized way. Blockchain is enabled by smart contracts to provide programmability; although promising, blockchains and smart contracts do not have access to the external world. Hence, blockchain oracles are introduced to resolve the blockchain connectability and expand the usability of smart contracts. This paper presented the livestock blockchain oracle (LBO) as a use case study for livestock DNA fingerprinting verification evaluated by real-world livestock datasets. The challenge was to connect the blockchain platform with these third-party DNA service providers holding privately owned sensitive data to provide a trusted and convenient way of verifying purchasable products. The designed blockchain oracle provided the connectability feature and employed the performance metrics of third-party data providers for finalizing the outcome. Results showed that the LBO is a responsive decentralized oracle.

References

[1] S. Ahmed and N. Ten Broek, "Blockchain could boost food security," *Nature*, vol. 550, no. 7674, pp. 43–43, 2017.

[2] D. Mingxiao, M. Xiaofeng, Z. Zhe, W. Xiangwei, and C. Qijun, "A review on consensus algorithm of blockchain," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 2567–2572.

[3] 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.

[4] S. Ellis, A. Juels, and S. Nazarov, "Chainlink a decentralized oracle network," https://link.smartcontract.com/whitepaper, 2017.

[5] 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.

[6] 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.

[7] H. Al Breiki, L. Al Qassem, K. Salah, M. Habib Ur Rehman, and D. Sevtinovic, "Decentralized access control for iot data using blockchain and trusted oracles," in *2019 IEEE International Conference on Industrial Internet (ICII)*. IEEE, 2019, pp. 248–257.

[8] "Web3j: Web3 java ethereum dapp api," https://github.com/web3j, 2021.

[9] M. Merlini, N. Veira, R. Berryhill, and A. Veneris, "On public decentralized ledger oracles via a paired-question protocol," in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE, 2019, pp. 337–344.

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

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

[12] Y. Cai, G. Fragkos, E. E. Tsiropoulou, and A. Veneris, "A truth-inducing sybil resistant decentralized blockchain oracle," in *2020 2nd Conference on Blockchain Research Applications for Innovative Networks and Services (BRAINS)*. Wiley, 2020, pp. 128–135.

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

[14] 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.

[15] TLSnotary, "Tlsnotary a mechanism for independently audited https sessions," https://tlsnotary.org/TLSNotary.pdf, 2014.

[16] J. He, R. Wang, W. Tsai, and E. Deng, "Sdfs: A scalable data feed service for smart contracts," in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2019, pp. 581–585.

[17] S. Woo, J. Song, and S. Park, "A distributed oracle using intel sgx for blockchain-based iot applications," *Sensors*, vol. 20, no. 9, p. 2725, 2020.

[18] S. Wang, H. Lu, X. Sun, Y. Yuan, and F. 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.
[19] Ethereum, https://ethereum.org/en/whitepaper/, 2020.
[20] "Javascript bindings for the solidity compiler," https://github.com/ethereum/solc-js, 2021.
[21] "Ganache," https://www.trufflesuite.com/ganache, 2021.
[22] "Performance regression in ganache-cli," https://github.com/trufflesuite/ganache-cli/issues/677, 2021.