Computers and Electrical Engineering 83 (2020) 106582

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Reliability analysis for blockchain oracles[☆]

Sin Kuang Lo[a,b,*], Xiwei Xu[a,b], Mark Staples[a,b], Lina Yao[b]

[a] Data61, CSIRO, Level 5, 13 Garden St, Eveleigh, NSW 2015, Australia
[b] School of Computer Science and Engineering, University of New South Wales, NSW 2052, Australia

ARTICLE INFO

Article history:
Received 29 June 2019
Revised 12 December 2019
Accepted 12 February 2020

Keywords:
Blockchain
Blockchain oracle
Reliability
Fault tree analysis

ABSTRACT

Blockchain is an emerging technology that is increasingly supporting economic-ally-critical systems. The execution environment of blockchain is isolated from the external world and thus requires "blockchain oracles": agents that fetch information from the external world. Blockchain is known to be highly reliable, but oracles are off-chain components that could be points of failure in whole blockchain-based systems. The reliability of blockchain oracles has yet to be investigated. In this paper, we propose a framework to compare and characterize existing blockchain oracles mechanisms from industry. Our approach for reliability modelling and architecture analysis of blockchain oracle systems uses Fault Tree Analysis. By calculating the reliability of oracles mechanisms, we can identify weak links that affect the overall reliability of a blockchain-based system.

1. Introduction

A blockchain is a decentralized, immutable digital ledger that keeps transaction data across a large network of nodes. Blockchain has the capability to disrupt existing business models and infrastructure in many sectors [1–3] by providing a platform for decentralizing trust for data. Blockchains can also provide decentralized trust for general computation, with so-called *smart contracts*, most notably in Ethereum[1].

With their increasing criticality, blockchain exchanges and platforms have also been increasingly attacked [4]. Attacks often target the weakest link of a system, as this requires the least effort [5]. These attacks have led to losses of hundreds of millions of dollars in total. As blockchain achieves wider adoption, we expect it to be used not just in economically-critical contexts, but also in safety-critical contexts such as pharmaceutical supply chains and IoT system. Blockchain-based systems must be reliable.

A blockchain oracle is a mechanism that fetches data from the external world to include it in the isolated execution environment of a blockchain. Blockchain oracles are generally off-chain components, so the reliability properties of blockchains do not apply to them. Blockchain oracles are needed to bridge blockchains and the external world because of unique characteristics of blockchain. Some kinds of data in the external world are inherently unable to be independently validated by multiple distributed parties, for example because the data has restricted access, or is transient sensor data. Oracles import

[☆] This paper is for CAEE special section SI-bcia. Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Prof. Khaled Salah.

[*] Corresponding author at: Data61, CSIRO, Level 5, 13 Garden St, Eveleigh, NSW 2015, Australia.

E-mail addresses: SinKuang.lo@data61.csiro.au (S.K. Lo), Xiwei.Xu@data61.csiro.au (X. Xu), Mark.Staples@data61.csiro.au (M. Staples), lina.yao@unsw.edu.au (L. Yao).

[1] https://github.com/ethereum/wiki/wiki/White-Paper.

https://doi.org/10.1016/j.compeleceng.2020.106582
0045-7906/© 2020 Elsevier Ltd. All rights reserved.

this kind of data as transactions into a blockchain. However, oracles are likely to have lower reliability than blockchain platforms, and this may impact the overall reliability of a blockchain-based system that relies on an oracle. The reliability of oracle mechanisms needs to be evaluated to assess the overall reliability of blockchain-based systems.

In this paper, we have identified existing active blockchain platforms with oracle mechanisms from industry, reviewed them, and compared their approaches to reliability. We propose an approach to analyse the reliability of oracle mechanisms with Fault Tree Analysis (FTA). In the approach, activity diagrams are first created that model user queries on external data. These activity diagrams are derived from white papers describing the oracle mechanisms, and are then transformed into a Fault Tree Diagram (FTD) for analysis. We calculate reliability of the oracle mechanisms and identify weak links that affect overall reliability of blockchain-based systems. Potential common causes of failure are discussed. Our contributions are:

- Investigation, characterization, and review of oracle mechanisms provided by existing blockchain platforms.
- An approach to model the reliability of oracle mechanisms by tailoring existing work on reliability evaluation.
- Quantitative and qualitative analysis to compare oracle mechanisms.

The rest of this paper is organized as follows. Section 2 gives a brief background of blockchain, blockchain oracles and FTA. Section 4 discusses characteristics of existing blockchain oracles. Section 5 describes on our proposed approach for reliability analysis of blockchain oracles. Section 6 discusses the results of our reliability calculations. Section 7 discusses reliability patterns for blockchain oracles and common causes of failures.

2. Background

2.1. Oracles

Many applications built on blockchain need to interact with other external systems. So, the validation of blockchain transactions might depend on states of those external systems. Blockchain oracles provide the required data from external systems to the blockchain, including for use in smart contracts. There are five types of oracles.[2] To preserve the deterministic validation of blocks, normally smart contracts can only access data previously stored on the blockchain, and cannot use external data. The use of oracles makes communication possible from the external world to the blockchain, for example by recording external data on the blockchain in transactions. A review of existing blockchain platforms with oracle services can be found in Section 4.

2.2. Reliability and architecture analysis

Reliability analysis allows software and system engineers to quantitatively assess hardware, software, and systems in term of probability of failure [6]. Reliability analysis is especially important to analyze potential risks to safety and economically critical assets. FTA is one of the most commonly-used methods for reliability analysis. Architecture analysis is the process of evaluating the fitness of an architecture for its intended purpose. Architecture analysis is usually conducted to qualitatively analyze software, to reason about the trade-offs of different design decisions.

3. Related work

Dependability research has previously been conducted on blockchain platforms, but not on the individual components of blockchain-based systems. Dependability and security are comprised of attributes like availability, reliability and integrity [7]. Availability of Bitcoin and Ethereum has been previously investigated [8], which found that write availability for transactions is low in comparison with read availability. This research also identified that the commit time of a block is highly variable and is impacted by network reordering.

Wan et al. investigated bug characteristics in open source blockchain projects and identified 10 bugs categories in blockchain systems [9]. Their results show that semantic bugs are the most common runtime bug category and the frequency distribution of bug types has a similar trend across blockchain projects. Security bugs take the longest median time to be fixed while performance bugs take the longest average time to be fixed.

Yasaweerasinghelage et al. proposed architecture modelling and simulation approach to predict the latency of blockchain-based systems [10]. The proposed model can be used for evaluation of design decision in blockchain-based systems such as number of confirmation blocks, which can impact write latency and write availability.

4. Review of blockchain oracles

4.1. Overall architecture

Fig. 1 depicts a generic overall architecture of various types of oracle mechanisms based on our investigation of existing oracle solutions. A generic oracle mechanism starts with a *requester* creating a *smart contract* (① in Fig. 1) specifying the

[2] https://blockchainhub.net/blockchain-oracles/.

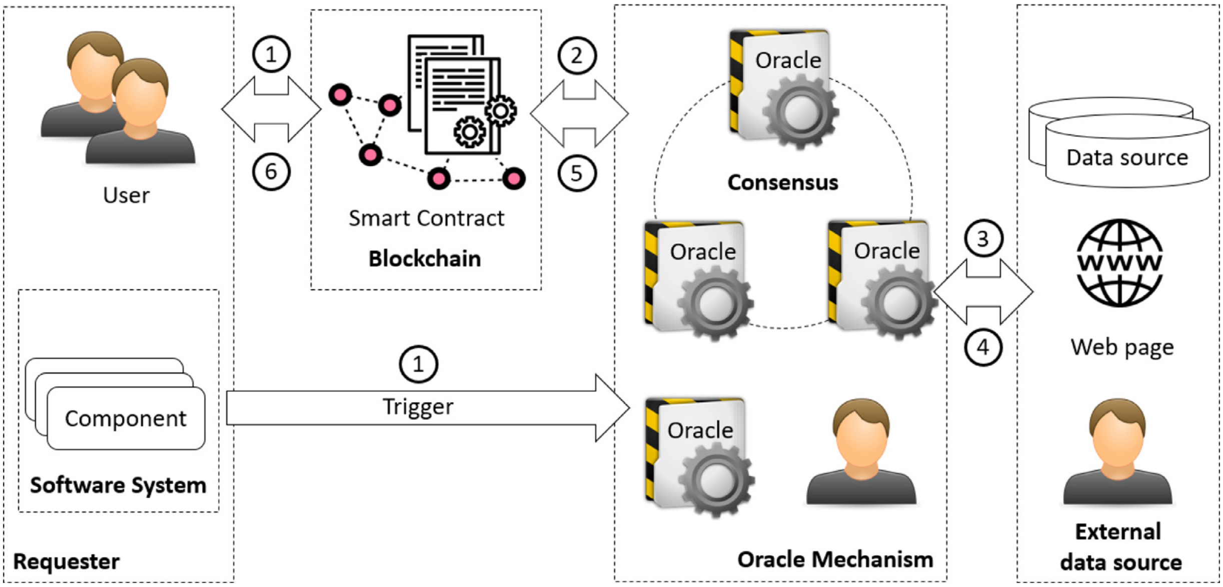

Fig. 1. Blockchain oracle mechanism architecture diagram.

Table 1
Summary of blockchain oracle mechanisms.

Platform	Oracle	Consensus	Reliability Feature(s)	Compatible platform	Data source(s)	Time interval	Type of oracles
Provable	single	N/A	TLS-Notary Proof	Bitcoin, Ethereum, Corda	single	fast	Provable contract
TownCrier	single	N/A	Intel SGX	Ethereum	single	fast	TownCrier contract
Corda	single	N/A	Intel SGX	Corda	single	fast	Corda code
MS Bletchley	multiple	N/A	Secure container, Intel SGX	Azure, AWS, Google	multiple	fast	off-chain code
ChainLink	multiple	N-of-M multi-signature	off-chain aggregation, Reputation	Bitcoin, Ethereum, Hyperledger	multiple	Slow	Reporter
Augur	multiple	Voting	Reputation, Dispute windows, Fork	Ethereum	multiple	Slow	Voter
Gnosis	multiple	Voting	Ultimate oracle & centralized oracle	Ethereum	multiple	Slow	Voter

data required to trigger the execution of the contract and deploy on the blockchain. The *requester* can be either a user or a component of a software system. In certain contexts, the requester could trigger the oracle directly (① Trigger in Fig. 1) to include a value into a blockchain to be used in the future.

Centralized oracles can automatically identify requirements specified by a *smart contract* (② in Fig. 1). A distributed oracle contain several redundant oracles that provide the same functionality to check the external state. Some oracles are humans with a blockchain account, who can manually enter oracle data and sign transactions.

Once an automated oracle has been deployed, it will communicate with its external data source, such as a physical sensor or web service, to collect the required data (③ and ④ in Fig. 1). Oracles will inject the data into the blockchain (⑤ in Fig. 1) and then the requester will be able to see this data in their execution of a smart contract (⑥ in Fig. 1).

Using an oracle introduces some issues:

- Oracles introduce a trusted third-party into the decentralised blockchain system. It needs to be trusted by all the participants involving in relevant transactions.
- Blockchain tramsactioms are immutable, but the external state used for oracle data may change.

4.2. Comparison of representative oracle solutions

We have selected seven currently-active blockchain platforms with oracle mechanisms. Their characteristics are summarized in Table 1. Provable, TownCrier, ChainLink are oracle service providers while Augur and Gnosis are prediction market leaders that leverage the capability of blockchain oracles. Provable, The columns of this table are discussed below.

4.2.1. Consensus

Consensus is how multiple oracles come to a final result to be submitted to the blockchain. For a single oracle mechanism, external information is fetched directly into the blockchain. For multiple distributed oracles, there are various consensus protocols used by different platforms to decide on the final result.

In ChainLink, a *K-out-of-M* threshold signature is used by multiple oracles to reach a consensus on the answer to be accepted. For example, a 3-out-of-5 signature scheme requires at least three or more oracles out of five oracles to sign on the same value for the value to be accepted as the answer.

Voting is used by Gnosis and Augur, where the oracles are human. After an oracle reports a result back to the blockchain, anyone with a blockchain account that does not agree with the answer reported by the oracle can dispute the result by reporting another value as a tentative answer by staking their token.

4.2.2. Reliability features

Reliability features are the distinctive attributes used by blockchain platforms to achieve reliability of their oracles. Provable introduces TLS-Notary[3] for fetching data from external data sources. TLS-Notary provides a cryptographic proof about data from a HTTPS secure site. TownCrier and Corda use Intel Software Guard Extensions (SGX) for hardware attestation to prevent unauthorized access outside of the SGX environment. TownCrier aims to establish a bridge between Ethereum and HTTPS-enabled websites. A concise piece of data (e.g. stock quotes) served to the blockchain is called a *datagram* in TownCrier. Corda uses *commands* for constantly-changing data and small-size data. Microsoft Bletchley has *cryplets* that work similarly to the datagrams of TownCrier. It utilizes Intel SGX hardware attestation to ensure a trusted connection for data to be submitted to the smart contract. ChainLink allows multiple oracles to fetch information from multiple data sources. Augur and Gnosis are both prediction marketplaces. Augur has a dispute time window that allows any user to disagree with the answer reported by an oracle.

Gnosis has three different oracle configurations, on chain oracle, centralized oracle and the ultimate oracle. 100 ETH is required to trigger the ultimate oracle if any user disagrees with the reported value. ETH is used on Gnosis because it is built on Ethereum. As of August 2018, one ETH is equivalent to $408 USD according to Coinmarketcap[4].

4.2.3. Data source(s)

Data source(s) are called by oracles to gather the information requested by the requester. A data source could be a static web page, physical sensor, component of a system or even input from a human. Despite having multiple data sources, some information such as the owner of a property from different resources might ultimately come from a single authorized source. For example information about city assets might only be authoritatively derived from the asset registry of the local city council.

4.2.4. Time interval

The time interval is the period between requesting data from an external data source until the data is returned back to the blockchain. Centralized oracle configurations have the shortest time interval, while multiple oracles require longer time intervals as data need to be aggregated.

4.2.5. Type of oracles

Provable, TownCrier, Corda use single oracle to fetch data from an external while MS Bletchley uses multiple oracles. ChainLink, Augur, and Gnosis use humans as oracles to report information back to the requester.

5. Reliability analysis framework

5.1. Approach overview

An overview of our proposed framework is shown in Fig. 2. White papers, official technical articles and official blogs of the selected blockchain platforms were studied to understand the characteristics of the oracle mechanisms and their features. A comparison between the features of seven selected oracle mechanisms can be found in Table 1.

UML activity diagrams (ADs) were manually generated for all the selected oracle mechanisms. All the generated ADs were cross-checked by another researcher to ensure they described the complete logic of the system. Any discrepancies were discussed and resolved. Examples of the ADs are shown in Fig. 3.

The ADs were then transformed into FTDs. The rules used to convert the ADs were implemented from [11]. An AD was transformed into to a success tree diagram and then inverted to form a FTD. As found out by Tiwari et al. [11], the AD can only be used to generate part of the FTD because not all the potential defects would be modeled in an AD. Therefore, all the identified potential faults of a system were manually mapped to the generated FTD afterward to form a complete FTD. Potential faults were extracted from relevant sources such as forums, github and publications. Minimum cut sets (MCS)

[3] https://tlsnotary.org/pagesigner.html.

[4] https://coinmarketcap.com/currencies/ethereum/.

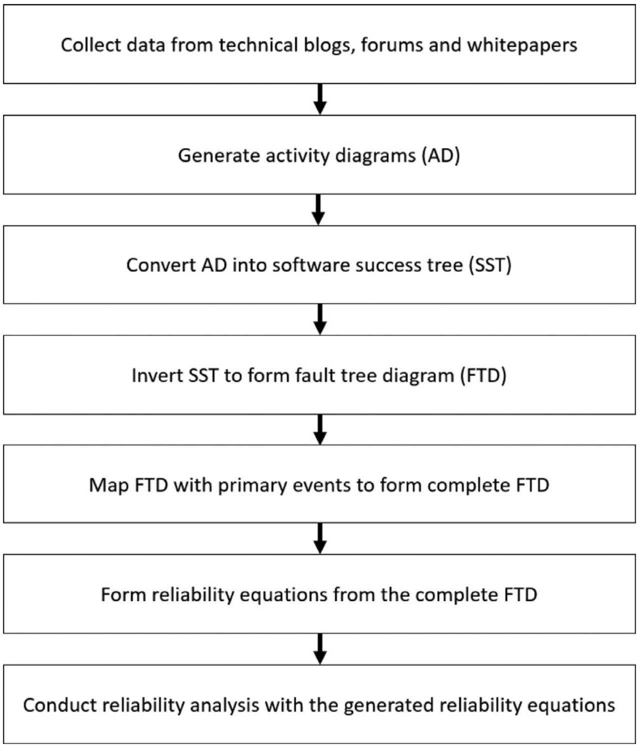
Collect data from technical blogs, forums and whitepapers
Generate activity diagrams (AD)
Convert AD into software success tree (SST)
Invert SST to form fault tree diagram (FTD)
Map FTD with primary events to form complete FTD
Form reliability equations from the complete FTD
Conduct reliability analysis with the generated reliability equations

Fig. 2. Overview approach to model reliability of blockchain oracles.

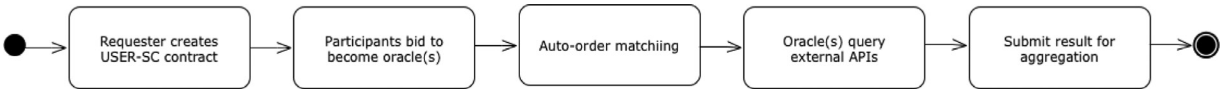
Requester creates USER-SC contract
Participants bid to become oracle(s)
Auto-order matchiing
Oracle(s) query external APIs
Submit result for aggregation

Fig. 3. ChainLink activity Diagrams.

Table 2
Summary of lower level errors.

Type of errors		Examples	Failure rate
	Smart contract error	Greedy contract Prodigal contract Suicidal contract	0.003678 [13]
	Server error	Server hacked Infrastructure error	0.001–0.005 (avg = 0.003)[7]
Human error	Simple task	Chose invalid data source	0.0005 [14]
	Routine task	Client side error Reporter not reporting	0.06 [14]
	Hard task	Update new features	0.1–0.25 [14]

were then generated from the FTDs in order to calculate the reliability of each oracle mechanism. A minimal cut set is the smallest list of events required to cause the top event to occur.

Both qualitative and quantitative analysis were conducted on the FTD. Common causes of failure are identified through qualitative analysis. For quantitative analysis, error rates of the events were used to calculate the reliability of the oracles systems. The error rates of all the events were identified from existing publications. Any unavailable error rate for an event in the FTD was substituted by similar traditional hardware component failure rates. All the events in the FTD were categorized into groups in Table 2 to determine the applicable error rate for similar events. Sensitivity analysis was also conducted to analyze the impact of the number of oracles on the reliability of the oracle(s) mechanism.

5.2. Model generation

5.2.1. Activity diagram generation

ADs are used to represent the logic of a single operation, use case or the flow of a business logic. An AD can be used to visualize the dynamic behavior of a system through the different flows of activities, such as parallel and concurrent activities.

An AD is used here to represent the flow of oracle processes from a request to query for external information, to the oracle fetching information off-chain, through to submitting the information back to the blockchain. The detail in the modeling of these processes was kept at the same level for all oracle mechanisms to allow comparable analysis of the oracle processes. Centralized oracle systems have a simpler AD compared with decentralized oracle systems. The AD generated for ChainLink is shown in Fig. 3.

5.2.2. Success tree diagram generation

The next step in the approach involves the generation of a Success Tree Diagram (STD) from the AD. A STD is a tree diagram which is used to analyze and identify the required elements to achieve the intended success. The STD presents the condition and elements to achieve the top event through a combination of various logic gates and basic events. An STD can be converted into a FTD for reliability and risk analysis.

From the AD shown in Fig. 3, we can observe that five activities are required for the ChainLink oracle to succeed in querying external information. Converting Fig. 3 gave us an STD with five events required to query the external information.

5.2.3. Fault tree diagram (FTD)

An FTD is the complement of a STD. FTDs are commonly used in engineering to analyze the potential risks to safety and economically-critical assets. They are also widely used in identifying risks of a software or system. FTDs are generally directed acyclic graphs, where a component's faults are modeled at the leaves of the graph. Logic gates represent how the faults propagate.

The events on the FTD are the complement from the STD, and gates are also swapped, e.g. an AND gate from the STD is changed into an OR gate in the FTD. A limitation of generating a tree diagram from an AD is that it might not cover all lower-level elements identified from forums, gitHub and publications[5] [6] that contribute to the top event. These lower-level elements are mapped onto the generated FTD to form the complete FTD, as shown in Fig. 4.

5.3. Reliability analysis

Reliability analysis is an essential part of designing, constructing and operating economically-critical technical systems [12]. Various methods and models have been created to support systematic analysis of the reliability and risk of a system and FTDs are one of the most commonly used models. A FTD generated in the previous step can be used to conduct both qualitative and quantitative analysis for reliability of blockchain oracle system.

A cut set (CS) is a unique set of events obtained from a FTD that is sufficient to cause the top event to happen. It provides a mechanism for probability calculations and also reveals the critical links in the system design [12]. A minimum CS (MCS) is the CS with minimum number of events that can cause the occurrence of the top event [12].

By using the complete ChainLink FTD in Fig. 4 as an example, we can form the MCS equation (Eq.) for the FTD as below:

$$T = P1 + P2 + P3 + P4 + P5 + K/N[P6 + P7 + P8 + P9 + P10]$$

ChainLink contains 10 single-component minimum cut sets. Any of the lower events from *P1* to *P10* is sufficient to cause the top event *T* to occur. Lower events from *P6* to *P10* are bounded by K-out-of-N gate, hence we applied the reliability Eq. for K-out-of-N to lower events *P6* to *P10*.

To calculate reliability of blockchain oracles using FTD, we show the reliability Eq. for a system and also reliability Eq. for K-out-of-N system.

$$R = \sum_{k}^{n} \frac{n!}{k!(n-k)!} (e^{-\lambda t})^k (1 - e^{-\lambda t})^{n-k} \tag{1}$$

Eq. 1 is the Eq. for *K-out-of-N system*

$$R_{ChainLinkoracle} = R_{P1} \times R_{P2} \times R_{Pn} \ldots \times R_{P10} \tag{2}$$

Eq. 2 below is obtained from the MCS for the ChainLink FTD

By substituting system reliability equation($R = e^{-\lambda t}$) and Eq. (1) into Eq. (2), the complete Eq. can be formed:

$$R_{P1\ldots P5} = e^{-\lambda_{P1} t} \times e^{-\lambda_{Pn} t} \ldots \times e^{-\lambda_{P5} t}$$

[5] https://github.com/provable-things/ethereum-api/issues/47.

[6] https://github.com/provable-things/ethereum-bridge/issues/33.

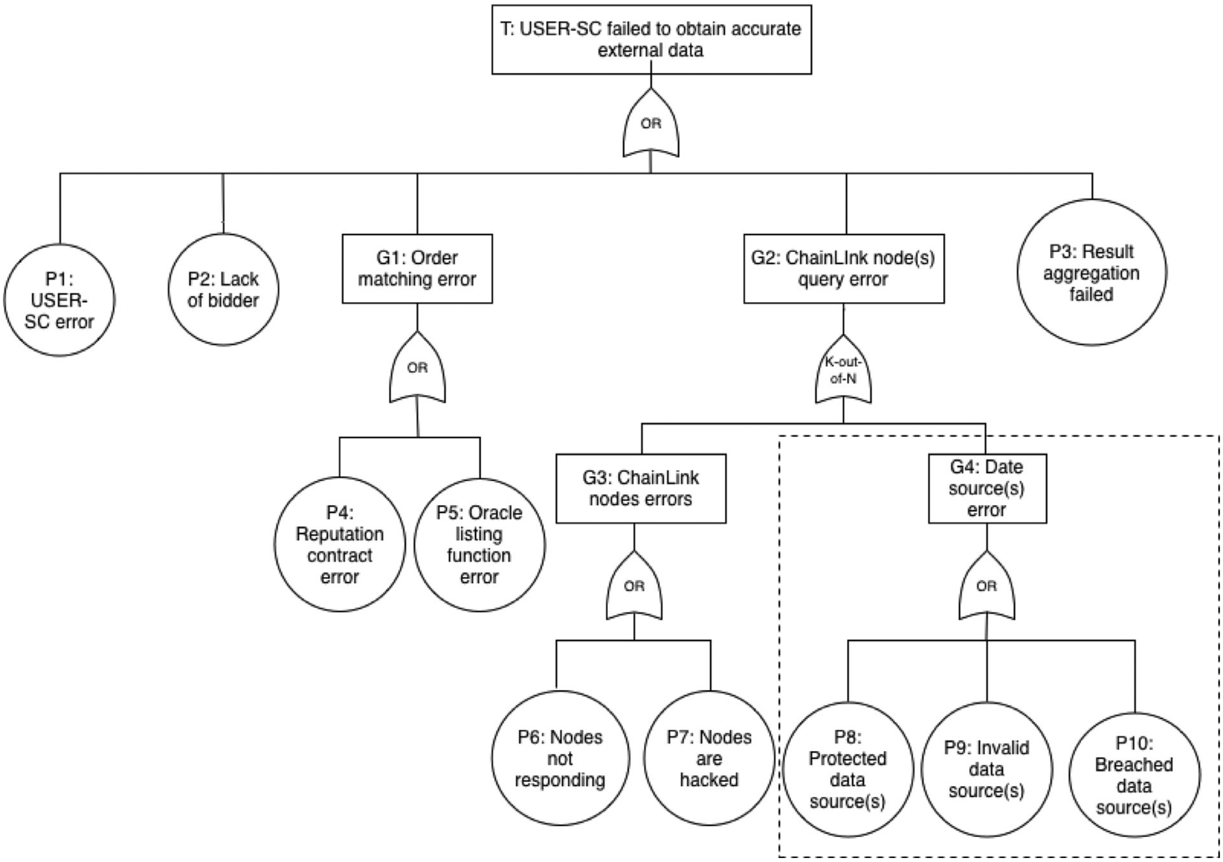
T: USER-SC failed to obtain accurate external data
OR
P1: USER-SC error
P2: Lack of bidder
G1: Order matching error
G2: ChainLInk node(s) query error
P3: Result aggregation failed
OR
K-out-of-N
P4: Reputation contract error
P5: Oracle listing function error
G3: ChainLink nodes errors
G4: Date source(s) error
OR
OR
P6: Nodes not responding
P7: Nodes are hacked
P8: Protected data source(s)
P9: Invalid data source(s)
P10: Breached data source(s)

Fig. 4. Complete ChainLink FTD after the mapping of lower events.

$$R_{P6\ldots P10} = \sum_{k}^{n} \frac{n!}{k!(n-k)!}(e^{-\lambda_{P6}t})^k(1-e^{-\lambda_{P6}t})^{n-k} \times \ldots$$
$$\times \sum_{k}^{n} \frac{n!}{k!(n-k)!}(e^{-\lambda_{P10}t})^k(1-e^{-\lambda_{P10}t})^{n-k}$$

Failure rates of the components are substituted into the Eq. to calculate the overall reliability of the oracle mechanism. Blockchain platforms and oracle services are too recent to have sufficient historical data for us to empirically calculate the failure rate of the components. Hence, to demonstrate the approach in this paper we have used failure rates reported in the literature for traditional software components. We also draw on some prior research on failure rates of smart contract. The values input to our model are tabulated in Table 2.

There are three types of major errors, which are *smart contract error, server error*, and *human error*. All issues related to data sources, such as server hacks and infrastructure errors, are grouped under the server failure category.[7] Human-related error can be categorized into three different types [14], including simplest possible task with failure rate of 0.0005, routine task with care needed with failure rate of 0.06 and complicated non-routine task with the failure rate of 0.1–0.25.

6. Results

The values of reliability of different oracles are shown in Table 3. Based on the values, Augur has the highest reliability, follow by MS Bletchley. Both TownCrier and Corda with the same reliability. The two platforms with lowest calculated reliability involve humans.

ChainLink designed its oracle solution with the need for humans to compete to become an oracle. Human error has the highest failure rate, hence contributes to the lower calculated value of ChainLink reliability. Augur also uses human oracles,

[7] https://forums.aws.amazon.com/thread.jspa?threadID=37676.

Table 3
Reliability of oracle(s) mechanism.

Platform	Reliability
Augur	0.9928
Ms Bletchley	0.9861
TownCrier	0.9840
Corda	0.9840
Provable	0.9810
ChainLink	0.9297
Gnosis	0.8837

but uses both a designated reporter and open reporters. This reduces the risk of not having anyone act as reporter, because there is the choice to choose a trusted oracle from a pre-defined list.

Augur allows disputations of the reported value by staking 2 times the amount of the no-show bond (amount of REP used in initial market creation). Gnosis also offers the same dispute windows but a fixed amount of 100 ETH is required in order to dispute the reported value. This discourages spurious disputes due to the high stake required. Gnosis network's participants might be reluctant to dispute an incorrect reported value because of the required high stake and thus reliability of Gnosis becomes lower than Augur.

Provable, Town Crier, Corda and MS Bletchley have a similar oracle mechanism. They all utilize trusted hardware to directly fetch information from an external trusted execution environment (TEE).

Provable offers TLS-Notary proof to validate the actual data submitted by an URL, but this serves as a single point of failure. TownCrier and Corda have a similar setup while MS Bletchley has multiple oracles that fetch data from multiple sources, resulting in higher calculated reliability compared with the other single oracle mechanisms.

Augur, Gnosis, MS Bletchley and ChainLink propose to have more than one oracle. Multiple oracles prevent single points of failure. Augur and Gnosis allow voting on correct answers by any participating node, and disputes on tentative answer. Any participant in a market has the ability to act as an oracle as long as they stake a result. MS Bletchley and ChainLink also have multiple oracles. ChainLink implements a K-out-of-N scheme, where the final consensus on the result will only be reached if a pre-determined K-out-of-N oracles agree on the reported value.

6.1. Sensitivity analysis

We conducted sensitivity analysis specifically on the K-out-of-N oracles scheme, as shown in Fig. 5.

By setting K as a constant to 1, the sensitivity analysis of N (blue curve) shows that the higher the N value, the lower the probability of failure and the higher the reliability of the oracle. We also conducted a K sensitivity analysis by fixing the number of N to 5 and change the number of K, the result (green curve) shows that the smaller the number of K, the higher the reliability. This higher the number of K, the more oracles are required to work correctly. Any K erroneous oracles will result in the failure of the whole system.

Finally, we performed a $N/2+1$ oracle sensitivity analysis. This is the scheme proposed in ChainLink's white paper. The result (red curve) shows that the higher the number N, the higher the reliability of the oracles services. N can only be an odd number because having an even number of oracles might result in deadlock if half vote for one branch and the other half vote for another branch.

Having more oracles reporting would increase the reliability of the services, but might incur a higher cost to be paid for all the oracles to work.

7. Discussion

7.1. Reliability patterns

During our study, we found that the designs of all oracle mechanisms are similar to various system fault-tolerance design patterns.

The active-active redundancy pattern is an approach to have a minimum of two nodes providing services simultaneously [15]. ChainLink and MS Bletchley oracle configurations are equivalent to the active-active redundancy pattern. Both blockchain platforms have multiple active oracles that fetch external data at the same time. MS Bletchley uses automated redundant oracles while ChainLink uses humans as redundant oracles. By having multiple active oracles fetch values into the blockchain, the risk is reduced from a single point of failure. Acquiring data by multiple oracles also improves the reliability of the final accepted value. The oracles are allowed to fetch the requested data from multiple external data sources, improving the likelihood of getting accurate external data.

The active-active redundancy reliability pattern has some disadvantages for cost and time. Data requesters may need to pay more to use multiple oracles to retrieve data and submit to a blockchain. Due to the need to aggregate collected data from multiple oracles and sources, a longer time frame is also required for all oracles to fetch values and aggregate the obtained values into a final value.

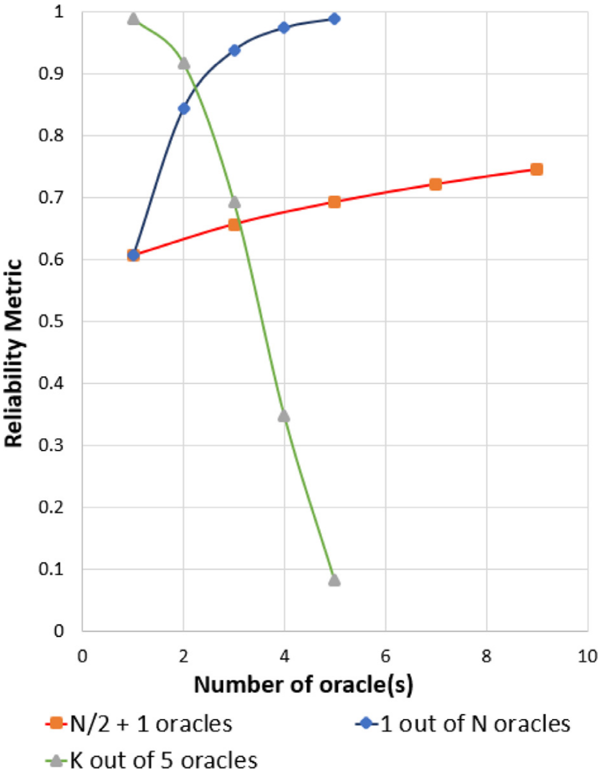

Fig. 5. Sensitivity analysis of K-out-N oracles on ChainLink.

MS Bletchley has a higher reliability because ChainLink oracles require human intervention. Participants in the network would need to bid to become an oracle. If no one is interested, the process will stuck after data is requested. Hence, the incentive scheme plays a very vital role to ensure involvement and interest from participants in the ChainLink network.

The active-passive redundant pattern has a minimum of two nodes, in which one acts as a standby server. The standby server will only take over if the active server is failing [15]. Augur and Gnosis configurations are similar to the active-passive redundancy pattern. In Augur and Gnosis, there is only one oracle reporting the value, so the initial reporting oracle is acting as the active server. The other participants in the network have the ability to dispute the tentative value if they find that the reported value is incorrect. The disputers act as a kind of passive server. Augur reliability is higher than Gnosis because for participants to dispute a reported value in Gnosis, the disputers need to pay 100 ETH to trigger the ultimate oracle. This economically discourages participants to rectify any misreported value due to the high stake required.

TownCrier and Corda use the input guard design pattern for fault containment with Intel SGX technology. The input guard design pattern stops the propagation of an error from the outside into the guarded component [16].

7.2. Common causes of failure

All oracles utilize methods to mitigate the risk of wrong information being used in the blockchain, which might cause incorrect execution of smart contracts. Nonetheless, the data source(s) can always be a common cause of failure. The dotted line box in Fig. 4 shows an example where common causes of failure might affect all the three data sources.

Consider the situation where there is a request for a weather forecast for Randwick, Sydney on a specific day. Multiple oracles will search for sources for a weather forecast from the internet and submit back to the requester. If the Randwick weather forecast is predicted by the Bureau of Meteorology and all sources derive the prediction from the Bureau, spoiled sensors or reports would impact all the sources. The result received by the oracle would still be valid from the operational point of view but the reliability of oracles in submitting accurate data would be affected. Hence, data collected from multiple web service data sources might be solely based on a single original data source from the physical world or another web service.

8. Conclusion

Blockchain oracles are a crucial component in expanding the capability of blockchain. We have selected, reviewed and characterized seven active blockchain platforms with oracle mechanisms according to their reliability features, consensus, and other characteristics. Our framework can serve as a reference on deciding suitable oracle mechanisms to fulfill different requirements. To assess reliability of blockchain-based systems using oracles, we tailored existing approaches for reliability calculations using FTDs to model and evaluate the reliability of the selected oracle platforms. We showed the calculation of reliability using this approach for the selected oracles based on representative failure rates of the components in the oracle mechanisms. The result shows that decentralized oracles are more reliable than centralized oracle mechanisms, and human-related faults are the main factor affecting the overall reliability of oracle mechanisms.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.compeleceng.2020.106582.

CRediT authorship contribution statement

Sin Kuang Lo: Conceptualization, Data curation, Writing - original draft. **Xiwei Xu:** Conceptualization, Data curation, Writing - original draft. **Mark Staples:** Conceptualization, Writing - original draft. **Lina Yao:** Conceptualization, Writing - original draft.

References

[1] Lo SK, Xu X, Chiam YK, Lu Q. Evaluating suitability of applying blockchain. In: ICECCS. IEEE; 2017. p. 158–61.
[2] Almadhoun R, Kadadha M, Alhemeiri M, Alshehhi M, Salah K. A user authentication scheme of IoT devices using blockchain-enabled fog nodes. In: 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA). IEEE; 2018. p. 1–8.
[3] Hasan HR, Salah K. Combating deepfake videos using blockchain and smart contracts. IEEE Access 2019;7:41596–606.
[4] Iqbal M, Matulevičius R. Blockchain-based application security risks: asystematic literature review. In: Proper HA, Stirna J, editors. Advanced information systems engineering workshops. Cham: Springer International Publishing; 2019. p. 176–88.
[5] Anderson R, Moore T. Information security economics–and beyond. In: Annual international cryptology conference. Springer; 2007. p. 68–91.
[6] Ohba M. Software reliability analysis models. IBM J Res Dev 1984;28(4):428–43.
[7] Avizienis A, Laprie J-C, Randell B, Landwehr C. Basic concepts and taxonomy of dependable and secure computing. IEEE Trans Depend Secure Comput 2004;1(1):11–33.
[8] Weber I, Gramoli V, Ponomarev A, Staples M, Holz R, Tran AB, Rimba P. On availability for blockchain-based systems. In: 2017 IEEE 36th Symposium on Reliable Distributed Systems (SRDS). IEEE; 2017. p. 64–73.
[9] Wan Z, Lo D, Xia X, Cai L. Bug characteristics in blockchain systems: alarge-scale empirical study. In: MSR; 2017. p. 413–24.
[10] Yasaweerasinghelage R, Staples M, Weber I. Predicting latency of blockchain-based systems using architectural modelling and simulation. In: ICSA 2017. IEEE; 2017. p. 253–6.
[11] Tiwari S, Gupta A. An approach to generate safety validation test cases from UML activity diagram. In: APSEC; 2013. p. 189–98.
[12] Dhillon BS. Engineering reliability: new techniques and applications. Technical Report; 1981.
[13] Nikolić I, Kolluri A, Sergey I, Saxena P, Hobor A. Finding the greedy, prodigal, and suicidal contracts at scale. In: Proceedings of the 34th annual computer security applications conference. In: ACSAC '18. New York, NY, USA: ACM; 2018. p. 653–63.
[14] Smith DJ. Reliability, maintainability and risk: practical methods for engineers. Butterworth-Heinemann; 2017.
[15] Saridakis T. A system of patterns for fault tolerance.. In: EuroPLoP; 2002. p. 535–82.
[16] Saridakis T. Design patterns for fault containment.. In: EuroPLoP; 2003. p. 493–520.

Sin Kuang Lo is currently pursuing a Ph.D. degree with Data61, CSIRO, Sydney, and the School of Computer Science and Engineering, UNSW. His current research interests include software architecture and blockchain, specifically on the role of blockchain within larger software systems.

Xiwei Xu is a Senior Research Scientist with the Architecture and Analytics Platforms Team at Data61, CSIRO, Sydney, and also a Conjoint Lecturer with the School of Computer Science and Engineering, UNSW. Her main research interests include software architecture and blockchain. She is the first author of the book Architecture for Blockchain Applications.

Mark Staples is a Senior Principal Researcher at Data61 and a Conjoint Associate Professor at UNSW with the School of Computer Science and Engineering. His research focus is in software engineering, software architecture, and blockchain. He is a co-author of the book Architecture for Blockchain Applications.

Lina Yao is a Senior Lecturer with the School of Computer Science and Engineering, University of New South Wales. Her research interest lies in Data Mining and Machine Learning applications with the focuses on recommender systems, human activity recognition, human-computer cooperations, Brain-Computer Interface and the Internet of Things.