Decision Support Model for Selecting the Optimal Blockchain Oracle Platform: An Evaluation of Key Factors

SABREEN AHMADJEE, Department of Cybersecurity, College of Computing, Umm Al Qura University, Makkah, Saudi Arabia and University of Birmingham, Birmingham, UK
CARLOS MERA-GÓMEZ, ESPOL Polytechnic University, Escuela Superior Politécnica del Litoral, ESPOL, Campus Gustavo Galindo, Guayaquil, Ecuador and School of Computer Science, University of Birmingham, Birmingham
SIAMAK FARSHIDI, Information Technology Group, University of Wageningen University & Research, Wageningen, Netherlands
RAMI BAHSOON, University of Birmingham, Birmingham, UK
RICK KAZMAN, University of Hawaii, Honolulu, Hawaii, USA

Smart contract-based applications are executed in a blockchain environment, and they cannot directly access data from external systems, which is required for the service provision of these applications. Instead, smart contracts use agents known as blockchain oracles to collect and provide data feeds to the contracts. The functionality and compatibility with smart contract applications need to be considered when selecting the best-fit oracle platform. As the number of oracle alternatives and their features increases, the decision-making process becomes increasingly complex. Selecting the wrong or sub-optimal oracle is costly and may lead to severe security risks. This article provides a decision support model for the oracle selection problem. The model supports smart contract decision-makers in selecting a secure, cost-effective, and feasible oracle platform for their applications. We interviewed oracle co-founders and smart contracts experts to refine and validate the decision model. Two real-world smart contract application case studies were used to evaluate the model. Our model prioritises and suggests more than one possible oracle platform based on the developer's required criteria, security assessment and cost analysis. Moreover, this guided decision model serves to reveal issues that may go unnoticed if done haphazardly, reduce decision-making efforts and provide a cost-effective solution.

CCS Concepts: • **Security and privacy** → **Software security engineering**; **Distributed systems security**; • **Software and its engineering** → **Software design engineering**;

Additional Key Words and Phrases: blockchain oracle, smart contracts, technical debt, MCDM

ACM Reference format:
Sabreen Ahmadjee, Carlos Mera-Gómez, Siamak Farshidi, Rami Bahsoon, and Rick Kazman. 2025. Decision Support Model for Selecting the Optimal Blockchain Oracle Platform: An Evaluation of Key Factors. *ACM Trans. Softw. Eng. Methodol.* 34, 1, Article 23 (January 2025), 35 pages.
https://doi.org/10.1145/3697011

Authors' Contact Information: Sabreen Ahmadjee (corresponding author), Department of Cybersecurity, College of Computing, Umm Al-Qura University, Makkah, Saudi Arabia and University of Birmingham, Birmingham, UK; e-mail: smahmadjee@uqu.edu.sa; Carlos Mera-Gómez, ESPOL Polytechnic University, Escuela Superior Politécnica del Litoral, ESPOL, Campus Gustavo Galindo, Guayaquil, Ecuador and School of Computer Science, University of Birmingham, Birmingham, UK; e-mail: cjmera@espol.edu.ec; Siamak Farshidi, Information Technology Group, Wageningen University & Research, Wageningen, Netherlands; e-mail: siamak.farshidi@wur.nl; Rami Bahsoon, School of Computer Science, University of Birmingham, Birmingham, UK; e-mail: r.bahsoon@cs.bham.ac.uk; Rick Kazman, Information Technology Management, University of Hawaii, Honolulu, Hawaii, USA; e-mail: kazman@hawaii.edu.

ACM 1557-7392/2025/1-ART23
https://doi.org/10.1145/3697011

1 Introduction

Blockchain-based smart contracts run in isolated environments to guarantee security and trustlessness, but some smart contract applications cannot function without access to off-chain data sources. These applications need a blockchain oracle component to enable the smart contracts interaction with real-world data and events. This component collects and validates data from a variety of external sources and transmits it to the blockchain. Examples of smart contract applications that benefit from the use of blockchain oracle platforms include **decentralised finance (DeFi)** [39], where oracles provide accurate price feeds for assets like cryptocurrencies and dynamic interest rate data; supply chain management [41], where oracles fetch real-time data from **Internet of Things (IoT)** devices to monitor product conditions and manage inventory levels; insurance [32], with oracles accessing weather data for automatic payout triggers and flight delay information for travel insurance policies; and gaming and gambling [18], where oracles supply random numbers for fairness and real-time sports scores for settling bets.

Numerous blockchain oracle platforms have emerged recently, such as Provable Things [10] and Chainlink [24], and are utilised in a variety of smart contract applications. As the number of available oracle platforms and their assessment criteria continues to grow, the selection of oracle alternatives available increases complexity and makes taking wrong or sub-optimal decisions more likely [25, 44]. These decisions become costly to fix because switching from one oracle platform to another requires uploading the smart contracts to the public blockchain, which is not free. For each deployment process, a specific amount of money needs to be paid [42]. Moreover, since these incorrect or unjustified oracle selections result from a failure to properly assess the available oracle platforms, they may also inadvertently lead to hidden **security risks (SRs)** [13].

This study provides a model to support decision-making when selecting blockchain oracles. Our decision model is built using the **multi-criteria decision-making (MCDM)** approach [59], which involves evaluating a collection of alternatives while considering a set of decision criteria. The model assists smart contract developers, blockchain project decision-makers and security software engineers in selecting an optimal oracle alternative to integrate into blockchain-based software. This model will inform them of the existing collection of oracle platforms and the relevant decision criteria and quality attributes they should consider when building this software category. Our model leverages the security technical debt metaphor to assess the ill consequences of sub-optimal selection that manifest as a debt that will accumulate interest over time [4]. Therefore, blockchain software engineers and decision-makers can assess each oracle alternative systematically and make better-informed decisions, considering the monetary cost of integrating and executing alternative options. The primary contributions of this work are:

- —A blockchain oracle decision support model, which is the first to guide blockchain software engineers and decision-makers with the systematic selection of a feasible, secure and cost-effective oracle for specific blockchain applications or systems.
- —A compilation of information on the features and quality attributes of eight cutting-edge blockchain oracle platforms to inform oracle selection decision-making systematically.
- —A leverage of the security technical debt metaphor to quantify the SRs of possible attacks on examined blockchain oracles, and a guide to discard solutions that can manifest into potentially costly and risky technical debts.

Our study is the first to propose an oracle decision model that assists smart contract developers in analysing the security of each alternative oracle by using well-informed decision-making. Different from previous research that investigated only a small subset of the features of blockchain oracles [5, 44], our research compiles an extensive set of blockchain oracles features to describe how they can

be mapped to quality attribute requirements, and calculates the monetary cost of integrating each of the available oracle options. Additionally, in contrast to previous efforts that described several attacks aimed against blockchain oracles [25], this work categorises and evaluates the attacks based on their likelihood and potential consequences, considering the domains of smart contracts in representative applications.

The remainder of the article is structured as follows. Section 2 introduces the necessary preliminary explanations of blockchain oracles and technical debt, while Section 3 provides an overview of our research approach. Section 4 presents the decision model for the oracle platform selection problem. Section 5 demonstrates two smart contract applications that were analysed to evaluate the effectiveness of the decision model, followed by an evaluation and a discussion of our results in Sections 6 and 7, respectively. Finally, Section 8 sets out the conclusions of our work.

2 Preliminaries

Smart contracts are self-executing computer programs that run automatically when certain criteria are satisfied. Many smart contract applications built on blockchain technology, such as Ethereum [66], need to interact with other external systems. Smart contracts may need data such as price feeds, weather data or even the generation of random numbers. However, smart contracts cannot fetch the required data since it is not possible to access resources outside of the blockchain environment [40].

Oracles can obtain and validate external data for smart contracts by using methods such as web APIs or market data feeds. The oracle initially monitors the blockchain network for off-chain data requests from users or smart contracts. It then interacts with the data source to find the required data and connects to that source to interface the blockchain with the data feed. Finally, the oracle provides data, and any associated proofs to the smart contract on the blockchain for consumption [15]. Therefore, the execution of the smart contract relies on the data retrieved from the data feed. Using an oracle introduces a trusted third party to the decentralised, trustless blockchain environment. This leads to a conflict between third-party oracles and the trustless execution of smart contracts regarding security, authenticity and trust.

Third-party or external dependency, which describes the use of any framework or software developed by external parties, is one of the causes that contribute to technical debt accumulation [45]. Technical debt is a metaphor devised to encapsulate how the value of software engineering decisions evolves. Evidence shows that a great deal of software depends on external parties to at least some extent, making it almost impossible to avoid incurring this debt [22]. This is because reliance on external sources might lead to invisible issues, such as security issues that might be transferred to the main product and make the entire system prone to attacks.

In blockchain oracle selection, technical debt can manifest because of compatibility issues between the chosen oracle platform and a specific application and/or the misconfiguration of the oracle, which can lead to security, availability, reliability or performance issues. Technical debt can also be caused by selecting an oracle without proper justification, e.g., if the features and services provided might not fully meet the use case's requirements. These issues could require re-selection, reconfiguration or redeployment, introducing additional costs and overheads.

In our model, we utilised the technical debt metaphor to assess the security issues that are rooted in the oracle platforms and can affect the whole smart contract application. Security technical debt has been shown to be an effective method for estimating the consequences to the business value of the vulnerabilities that are being exploited [36]. Steep accumulation of interest can be avoided if the debt is analysed at an early design stage [4]. Leveraging technical debt in our model benefits decision-making as it quantifies the impact of security issues over time, encouraging designers to

slow down when selecting an oracle. This allows them to select a secure and optimal oracle for their smart contract applications.

2.1 Hypothetical Scenario: Inadequate Oracle in a Smart Contract

In the DeFi ecosystem, smart contracts often rely on price oracles to determine the value of assets for various functions, such as liquidations, lending and trading. An inadequate oracle choice in this context can have severe consequences.

Let's consider a decentralised lending protocol called 'LendX' and two different oracle platforms (i) *Oracle Platform A*: This oracle platform is well-established and has a reputation for providing accurate and timely price data. It is widely used in the DeFi space; and (ii) *Oracle Platform B*: This oracle platform is relatively new and unproven but offers lower fees.

LendX, aiming to minimise costs, decides to integrate Oracle Provider B into its lending platform to save on fees and assuming it would perform adequately. LendX launched a lending platform where users could deposit collateral and borrow assets based on the value of their collateral. However, shortly after the platform's launch, Oracle Platform B experienced a technical issue that caused a delay in price updates for one of the assets used as collateral. During this delay, the price of the collateral asset plummeted on external markets due to a flash crash.

Because LendX's smart contracts relied on Oracle Platform B, they continued to use outdated and inaccurate price data. As a result, the smart contracts failed to trigger timely liquidations of under-collateralised positions, causing LendX to incur significant losses. Additionally, borrowers who used the depreciated asset as collateral faced the risk of having their entire collateral wiped out, which created substantial financial instability for Project X and its users. Moreover, the incident eroded trust in Project X's platform and the DeFi ecosystem in general, as users questioned the reliability of DeFi projects.

In this example, the choice of Oracle Platform B, which seemed cost-effective at the time, led to security vulnerabilities and adverse outcomes. It illustrates the importance of carefully selecting an oracle platform with a track record of reliability and suitable to the decentralised application.

3 Research Approach

The number of smart contracts oracle platforms and their features has recently expanded, complicating the decision-making process when selecting them. Making optimal decisions calls for smart contract developers to recognise the existing collection of alternatives and understand their related features so that it is possible to compare and assess the various platforms. MCDM techniques can be leveraged to rank and prioritise alternative options based on the required features and their relative importance. In general, MCDM techniques involve six steps [43] including (1) determining the objective; (2) determining various alternatives; (3) determining the criteria; (4) determining the weighting approach; (5) applying the aggregate approach; and (6) using the aggregate findings to make decisions.

MCDM processes can be executed using a variety of mathematical approaches, which are chosen based on the nature of the problem and the amount of complexity ascribed to the decision-making process. The decision model proposed in this study for Oracle platform selection is adapted from the MCDM framework [26], which was originally developed for modelling MCDM problems in software production. The framework has undergone testing and evaluation in various software engineering domains, including the Blockchain platform [29], Decentralised Autonomous Organisation platform [9] and Business Process Modelling Language [31] selection problems.

In this study, we extend the security technical debt equation proposed in our previous study [4] to estimate the security consequences and quantify the security debt interest. Our approach assists developers with recognising the SRs and uncertainties associated with each alternative and

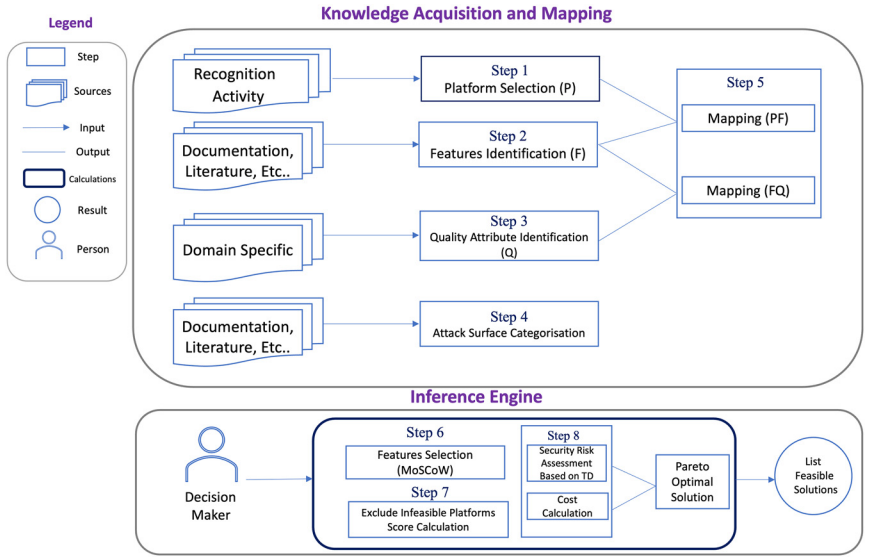

Fig. 1. Decision model structure for oracle selection.

supports them in selecting a secure solution. In addition to security, we leverage **multi-objective optimisation (MOO)** techniques to analyse the cost of deploying and executing a contract. These techniques produce a Pareto front optimisation, in which all the solutions are Pareto optimal [14]. This suggests that no single solution is best for all purposes but rather a collection of alternatives with various tradeoffs among the competing goals.

Our systematic decision model is grounded in a document analysis of an extensive review of the main sources of knowledge, i.e., the literature concerning smart contracts and oracles. Expert interviews were another source of knowledge used to refine and validate the decision model. We interviewed five experts (three oracle co-founders and two domain experts). The interviews followed a semi-structured interview protocol and lasted between 30 and 45 minutes. We demonstrate how the decision model can be applied using two exploratory theory-testing case studies. We support the decision model with a stability and sensitivity analysis, which checks the extent to which the final decision can be affected when altering critical factors.

4 MCDM For Blockchain Oracle Platform Selection

This section discusses the eight-step process used to formulate the blockchain oracle selection decision support model. Three of the steps (namely five, six and seven) were based on a well-known decision support system proposed by Farshidi et al. [28]. Adapting these steps in our context is feasible as they are applied to a decision model related to the blockchain platform selection problem [29]. However, we contribute the other five steps. Detailed explanations of each step are provided in the following subsections. First, the process of extracting, organising and arranging the information from various sources is explained in Section 4.1. Then in Section 4.2, we elaborate the aggregation equations that transform the criteria into a set of results to find the best-fit candidate oracle platforms. We also justify how the MOO technique can be adapted to find secure and cost-effective solutions. Figure 1 illustrates the main steps of our systematic model.

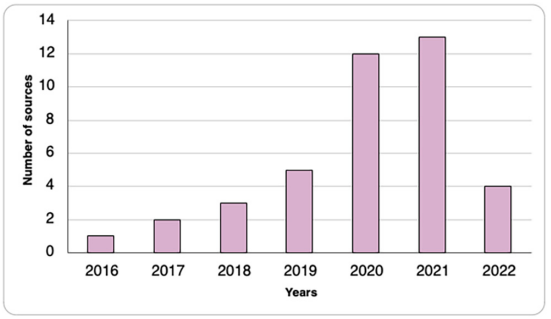

Fig. 2. Distribution of chosen studies throughout the document analysis phase.

4.1 Knowledge Acquisition and Mapping

The primary sources of information used to formulate the steps related to the knowledge acquisition and organisation phase were expert interviews, online documentation and literature studies. We investigated the literature using the following search query: '(Blockchain OR Smart Contract) AND (Oracle OR Oracles OR Decentralised Oracle OR Data feed)'. Due to the novelty of blockchain oracles and the industry's rapid growth and development, we only found a limited number of academic articles.

We expanded our knowledge acquisition process to include grey literature sources, which resulted in the extraction of 40 relevant documents. At the time of writing, around 40% of the results found are academic papers, 25% are blogs and forum articles, 18% are white papers and documentation of the platforms themselves and 10% are collections of videos presenting platforms functionalities or developers interviews explaining platforms features. Despite the data for the year 2022 is based on the available information up to the time of writing, we observed that the investigation of blockchain oracles has gained more attention in the past 2 years, as shown in Figure 2.

We implemented a document analysis procedure [20] for reviewing and examining the selected sources of knowledge; the procedure assisted us to derive information and insights regarding the oracles. As part of the document analysis step, we took a systematic approach to populate an extraction form. This involved a review of each document to identify and extract relevant information. Specifically, we examined the content of the documents, focusing on the following criteria to compile information comprehensibly: oracle platforms, oracle features, related quality attributes, security flaws and potential attacks. Information meeting these criteria was then systematically recorded in the extraction form to ensure accuracy and consistency in data collection. The information extraction process involved two authors working collaboratively to extract data. Any conflicts that arose during this process were resolved through discussion and consensus between these two authors. Subsequently, the extracted data underwent validation by the remaining authors to ensure accuracy and reliability. The knowledge acquisition and mapping comprises the first five-steps of our model as follows.

4.1.1 Step One (Platform Selection). In addition to the extracted information from the document analysis phase, we inspected GitHub repositories to identify oracle alternatives. At the time of writing, we have identified 22 oracle platforms. All platforms and their sources are publicly available.[1] Popularity is a key factor that we used to measure the suitability of these platforms for inclusion in our decision model. Thus, two metrics are used to assess the popularity of each oracle platform: *recognition* and *activity*.

[1] https://bitbucket.org/Smart_Contract/oracle_dsm/src/master/

Recognition refers to the degree of acceptance of the oracle platform by developers and others. The recognition indicators that we used are the number of forks, the total number of commits and the number of stars in GitHub. GitHub forks denote the number of people who have participated (or want to participate) in the oracle; the total number of commits in GitHub denotes the development rate of the oracle; and the number of stars in GitHub denotes the number of developers who like the oracle.

Activity tracks the extent to which developers and others use a particular platform. The oracle is no longer popular when developers stop updating and enhancing it, or people stop talking about it. As an indicator of activity, we use the number of commits in GitHub in the previous month to express the development of the oracle platform since the previous month. The number of Forks is also an indicator of the development of the oracle platform. Additionally, we use the Keywords Everywhere tool [37] to find the monthly search volume of the GitHub page and the main Web page of the oracle platform if it exists. We have made the indicator values of each metric public for interested readers [2].

Since this study is primarily concerned with smart contracts that run on the Ethereum blockchain, we only considered the oracle platforms compatible with Ethereum. As a result, based on *recognition*, *activity metrics* and *compatibility with Ethereum*, we ended up with eight oracle platforms which are:

(1) *Provable Things (P1)* [10], known also as Oraclize, is a centralised blockchain oracle service that facilitates data communication between smart contracts and external sources. It acts as an intermediary, enabling smart contracts to query data from off-chain sources like web APIs and IoT devices. Provable uses cryptographic techniques to ensure data authenticity and integrity, making it tamper-proof. It often employs a callback mechanism to return the data to the requesting smart contract, providing a dependable way for applications to access real-world data while maintaining trust and security on the blockchain.
(2) *Town Crier (P2)* [70] is a centralised blockchain oracle solution developed by Cornell University researchers. Town Crier utilises Intel's **software guard extensions (SGX)** technology to securely retrieve and deliver external data to smart contracts while preserving confidentiality. This hardware-based approach ensures data privacy and integrity, making Town Crier suitable for applications requiring high levels of privacy, such as confidential information retrieval.
(3) *Chainlink (P3)* [24] is a semi-decentralised oracle network known for its robust and flexible architecture. It operates through a network of independent node operators who fetch, verify and deliver external data to smart contracts. Oracle selection is decentralised and trustless, considering factors like reputation and collateral. Chainlink uses cryptographic signatures for data verification, ensuring the accuracy and integrity of the data, making Chainlink is a popular choice as a blockchain oracle solution.
(4) *Witnet (P4)* [21], a semi-decentralised oracle network that uses a network of nodes to retrieve, validate and deliver external data to smart contracts. Witnet uses a network of nodes to fetch, validate and deliver data. It employs a three-step process, including data request, reporting and data consensus, to ensure data accuracy. It allows developers to create custom data requests for their smart contracts, making it versatile for various use cases, including finance (DeFi) and prediction markets.
(5) *Tellor (P5)* [16] is a fully decentralised blockchain oracle that employs a unique proof-of-work-based consensus mechanism, where miners compete to fetch and submit data to the blockchain. The most accurate data submissions are rewarded with native tokens (TRB). This incentivised system encourages data accuracy and integrity. Tellor's on-chain data storage

and dispute resolution mechanisms further enhance data reliability. By combining economic incentives with cryptographic security, Tellor ensures that blockchain applications have access to trustworthy external data, suitable for various use cases.

(6) *Paralink (P6)* [49] is a cross-chain oracle platform that uses a unique consensus mechanism that combines on-chain and off-chain computation to ensure data accuracy. Paralink allows developers to create custom oracle scripts, making it highly flexible for various use cases. Its cross-chain capabilities enable data delivery to multiple blockchain platforms, enhancing interoperability for decentralised applications.
(7) *BandChain (P7)* [8] is a cross-chain oracle platform that offers fast and customisable data feeds for smart contracts. It uses a delegated proof-of-stake consensus mechanism to aggregate and deliver data from multiple sources. BandChain provides developers with the flexibility to create custom oracle scripts and data request templates, making it suitable for a wide range of applications. Its speed and scalability are designed to support real-time data needs and it is suitable for applications requiring high-throughput data updates, such as DeFi and gaming.
(8) *iExec (P8)* [35] is a decentralised cloud computing platform that offers a marketplace for computing resources, including off-chain computation and oracle services. While primarily known for its computing capabilities, iExec also supports oracles to facilitate the integration of real-world data into smart contracts. Through its decentralised marketplace, users can access a wide range of computing and data services, making it suitable for complex computations and data-intensive applications on the blockchain.

4.1.2 Step Two (Features Identification). We performed an iterative content analysis method [12] to organise the information extracted from the interviews and the document analysis into categories related to oracle features. A series of well-defined steps were followed to analyse and organise the information comprehensively.

(1) *Data Collection*: The initial phase involved the meticulous collection of qualitative data from interviews and document analysis. This encompassed various forms of data, such as transcripts, notes and pertinent documents, relevant to the study of oracle features. Subsequently, a thorough process of data familiarisation commenced. This stage required an in-depth review of the gathered material from interviews and documents, enabling a holistic grasp of the data's essence.
(2) *Initial Categories*: Initial categories or codes were systematically formulated to encapsulate the salient themes, topics and concepts pertaining to oracle features. These initial categories aimed to encompass the most conspicuous patterns present in the data.
(3) *Coding*: The formulated initial categories or codes were then diligently applied to the collected data. When encountering information within interviews or documents that pertained to specific oracle features, it was methodically categorised within the established decision model.
(4) *Iterative Process*: The categories continued to evolve as new concepts were identified in the sources. The evolving nature of the data could unveil new concepts or variations, prompting the creation of fresh categories or adjustments to existing ones.

Performing content analysis allowed us to identify 37 features, which were then categorised into 13 categories. Our sources and extraction forms are publicly available [2]. Even though our model includes a large number of features—37 features—it allows decision-makers to adjust the selection to their own needs, allowing analysts to focus on the most important features that have the potential to meet the quality attributes of their blockchain-based application.

We briefly explain each category and its related features to prevent semantic mismatches throughout the oracle platform selection process.

(1) *Type of Oracle* refers to the type of entities that provide data from an external source to a smart contract. An oracle can be either centralised, semi-decentralised or fully decentralised. A centralised oracle relies on a central authority with complete control over the data being provided to the smart contracts. A semi-decentralised oracle relies on a decentralised network of nodes, or a pre-determined number of nodes in some oracle, to provide data. Still, they also rely on a central authority or centralised infrastructure. A fully decentralised oracle relies on an undetermined number of decentralised nodes to provide data, and it does not depend on any central authority to control or manipulate the data.
(2) *Type of Data Source* refers to the data sources that oracles use to collect the data they send to smart contracts. A single-source oracle uses just one data source, while a multi-source oracle uses numerous data sources.
(3) *Data Validation Mechanism* refers to the process through which oracles verify that the data submitted to the contract are accurate. Some oracles utilise a consensus-based solution for data validation. Other platforms rely on a third-party approach and assume the data source is trustworthy. A trusted execution environment is also used by a set of oracles to ensure that data are processed and protected in a secure environment.
(4) *Integration Methods* refer to the mechanisms for connecting an oracle to a blockchain to provide data. Delivering the data to the contract can be completely off-chain, on-chain or hybrid.
(5) *Encryption Methods* are used to ensure confidentiality when data are sent from external data sources to the oracles.
(6) *Data Feeders Selection Method* ensures that the correct data are sent to the blockchain by only including legitimate data feeders and excluding malicious ones. This technique can be collateral-based (e.g., staking) or reputation-based. In the collateral-based approach, staking refers to a process where data feeders or validators are required to lock up a certain amount of cryptocurrency tokens as collateral to participate in the data feeding or validation process within a blockchain network. Staking is commonly used as a mechanism to encourage honest and accurate data reporting while discouraging malicious or incorrect data submissions. In the reputation-based approach, a reputation contract is generated to track the accuracy of the data provided by each feeder. This helps in picking the best data feeders from all those available.
(7) *Aggregation Mechanism* combines several data inputs into a single output. The output quality is determined by the data feeders chosen and by the aggregate method employed. Mean, median and mode are the three most important aggregation statistics. Weights can be used in the computation process to enhance the quality of simple statistics such as the median and mode.
(8) *Dispute Resolution* is designed to ensure the quality of the result and to provide stakeholders with an opportunity to correct potential errors. Several resolution mechanisms are available such as staking, voting and statistical approaches. Staking-based resolution incentivises the discovery of incorrect outcomes. Staking-based resolution often incorporates a selection mechanism that determines which data feeders are chosen to provide data for a specific query or request. The probability of being selected is proportional to the amount of tokens staked. Participants who have staked more tokens have a higher chance of being selected. Staking creates incentives for data feeders to provide accurate and reliable data. Participants have a vested interest in the integrity of the network because they risk losing their staked tokens if

they submit incorrect or malicious data. This aligns their incentives with the accuracy of the information they provide. In the case of a voting-based resolution, any token holder can vote on disputed data, and the decision is not limited to the data feeders. In a statistical approach, the median or the mode of the values is calculated from a set of values proposed by different feeders. After the data have been aggregated, statistical methods can be applied automatically and almost instantaneously. Voting, on the other hand, involves human judgement, which may result in more accurate results in complex situations.

(9) *Incorrect Data* can be either corrected or reverted. The new, uncontested value will be reflected in the corrected data. Reversion means the result will be nullified, and the system will have to start over to obtain a new outcome.

(10) *Incentive Schemes* are intended to encourage feeder nodes to be truthful and provide correct results to maximise their profits. There are several examples of incentive schemes commonly used to motivate data feeders in blockchain oracle systems, including, *token rewards*, where data feeders can be rewarded with tokens or cryptocurrencies native to the blockchain network for their data contributions; *reputation-based systems*, where the systems assign reputation scores or rankings to data feeders based on their historical performance, accuracy and reliability. Data feeders with a strong reputation have a higher chance of receiving rewards or being selected for data provision tasks, thus motivating them to maintain a good track record.

(11) *Punishment Methods*, on the other hand, apply appropriate punishments to feeders if the data they provide are shown to be incorrect. These may include banning, which refers to temporarily or permanently excluding the feeders; slashing, which refers to reducing the portion of the stack; or suffering a reputation loss that affects the standing or trustworthiness of the feeders within the network.

(12) *Native Tokens* are included with most oracle platforms. For example, Tellor has the TRB token, whereas other solutions, such as Provable, have not introduced a new token.

(13) *Using a Native Token* in some oracle platforms such as Chainlink requires users to use their token to request external data, while in the case of other platforms such as Witnet, users are allowed to use Ether for the requesting transaction.

4.1.3 Step Three (Quality Attributes Identification). As this study intends to assist in selecting secure and cost-efficient oracle platforms, we identified a set of security quality attributes that relate to blockchain oracles using the following sources: (i) the extracted form that was derived in the document analysis phase, where several quality attributes related to the blockchain oracle were determined; (ii) the NISTIR 8202 report [68] that presents information about blockchain technology; and (iii) the ISO/TR 23455:2019 document [1] that provides a detailed overview about smart contracts. The following security attributes were found in these sources: confidentiality (C), integrity (I), availability (A), **non-repudiation (NR)**, authenticity (Auth), trustlessness (Trust) and transparency (T). In addition to the security attributes, other criteria, such as cost efficiency, **low cost (LC)**, **low transactional financial cost (LTFC)**, **low latency (LL)** and accessibility (Acc), are also considered. The oracles' features were analysed based on their impact on the quality attributes that had been determined.

4.1.4 Step Four (Attack Surface Categorisation). In addition to the information acquired through the document analysis and the extracted information from the expert interviews, we use the categories that we developed in the study by Ahmadjee et al. [3] in which attacks that targeted centralised and decentralised blockchain oracles were identified and classified. We use this classification to categorise the attack surfaces. We have identified 12 distinct attacks that might target the oracle. Table 3 offers a brief explanation of each attack and shows which type of oracle each of

Table 1. A Sample of the Mapping between the Boolean Oracle PF

Boolean Oracle Criteria and Platform	P1	P2	P3	P4	P5	P6	P7	P8
Type of Data Feeder								
Centralised	1	1	0	0	0	0	0	0
Semi-decentralised	0	0	1	1	0	1	1	0
Fully decentralised	0	0	0	0	1	0	0	1
Type of Data Source								
Single	1	1	1	1	1	1	1	1
Multiple	0	1	1	1	1	1	1	1
Encryption Method								
Symmetric encryption	0	0	0	0	0	0	0	0
Asymmetric encryption	1	1	1	0	0	0	0	1

them targets. For example, front-running attacks [13] target both centralised and decentralised oracles. Other attacks, such as Sybil attacks [47], are only applicable when a decentralised oracle is in use, while other attacks, such as flash loan attacks [13] and SGX attacks [71], target centralised oracles. A 'flash loan' is a transaction in which a borrower takes out a loan without collateral, uses the money to complete specific tasks, and then repays the initial loan at the end of the transaction. Otherwise, the transaction fails if the original loan is not repaid at the end of the transaction.

4.1.5 Step Five (Features and Platforms (PF) Mapping). This step concerns performing the mapping between the set of Features (F) and the set of Platforms (P), and between the set of Qualities (Q) and the set of Features (F) during the knowledge acquisition and organisation stage. To apply the mapping, let $P = \{p1, p2, ..., p|Platforms|\}$ be a set of blockchain oracle platforms in the market (i.e., Chainlink and BandChain) and $F = \{f1, f2, ..., f|Features|\}$ be a set of features (i.e., Integration Methods and Incentive Schemes) of the oracle platforms. Each $p \in P$ supports a subset of features in the set F. The main objective is to select the most secure and cost-effective oracle platform p that supports the required oracle features.

According to the argument put forward by Farshidi et al. [28], the main sources of knowledge in this step could be the documentation of alternatives, literature studies and experts' knowledge. Thus, we determined the (PF) mapping between the sets of Platforms and Features based on the information extracted from the sources of knowledge and expert interviews. PF mapping is accomplished using a Boolean adjacency matrix ($Feature \times Platform \rightarrow \{0, 1\}$). $PF(p, f) = 0$ means that the oracle platform p does not support the oracle feature f, whereas $PF(p, f) = 1$ means that the platform employs the feature. For example, Provable, Town Crier and Chainlink support the Encryption Method feature as they can apply asymmetric encryption to the transmitted data, while the other platforms do not. Table 1 represents a sample of PF mapping. In the Appendix, Table A1 demonstrates the full mapping.

The (FQ) mapping between the set of Features and the set of Qualities is determined based on the extracted information form derived from the selected sources of knowledge, expert interviews and the authors' expertise. Their expertise is assessed in the fields of software architecture evaluation, quality attribute and tradeoff analysis, blockchain and security. A Boolean adjacency matrix is used in FQ mapping ($Quality \times Feature \rightarrow Boolean$) which means that each feature either provides or does not provide a specific quality attribute. To reduce inherent biases in the mapping process, there are two authors worked separately to map the qualities with the related features. Then, using Equation (1), we calculated Cohen's kappa (k) [56], which is a statistical approach to assess the

Table 2. A Sample of the Mapping between the Boolean Oracle Features and Quality Attributes

Boolean Oracle Criteria and Attribute	C	I	A	NR	Auth	Trust	T	LL	LC	LTFC	Acc
Type of Data Feeder											
Centralised	0	0	0	0	0	0	0	1	0	0	0
Semi-decentralised	0	0	1	0	0	0	0	0	0	0	0
Fully decentralised	0	1	1	0	0	1	0	0	0	0	0
Type of Data Source											
Single	0	0	0	0	0	0	0	1	0	0	0
Multiple	0	1	1	0	0	1	0	0	0	0	0
Integration Methods											
On-chain	0	1	1	0	0	0	1	0	0	0	0
Off-chain	0	0	0	0	0	0	0	0	1	0	0
Hybrid	0	1	0	0	0	1	1	0	1	0	0

agreement between two authors deciding on the same issue, as follows:

$$k = (P_o - P_e)/(1 - P_e), \tag{1}$$

where P_o is the probability of the observed agreement, and P_e is the probability of random agreement. Based on our two different mappings, we got a P_o of 0.96, and a P_e of 0.76 which produced a k of 0.83 indicating near-perfect agreement. Finally, to reach a consensus and address any disagreement, we convened a group discussion involving three contributors: (i) A professor of specialising in distributed and autonomous software engineering; (ii) A Ph.D. student renowned for their expertise in cybersecurity and software engineering; (iii) A university professor with a remarkable 11-year background in the field of cloud software engineering, boasting extensive industrial experience. Additionally, valuable feedback on the mapping was supplied by one of professors and industrial research co-author, recognised globally for their seminal work and creation of influential methods and tools for architecture analysis, tradeoffs analysis and widely employed in the software industry. We have used Cohen's kappa to check for disagreements and discrepancies in scores to elevate subjectivity and strive for consistency. The FQ mapping does not affect the final results, but they do assist in the qualitative assessment and profiling of the alternative solutions under consideration.

Table 2 shows a sample of a final FQ mapping. Zero has been assigned if the oracle feature is prone to security threats that would likely have an effect on the quality attribute(s) that are of interest. Otherwise, a value of 1 has been assigned if the likelihood of threat is not evidenced in the review of the source of knowledge or further discussions on the case. For instance, a centralised oracle introduces a single point of failure threat as it is prone to denial of service attacks, which might affect oracle availability [25]. A decentralised oracle resolves these issues as it contains several redundant oracle servers, which leads to better availability. Additionally, as the figure shows, the features provided by oracles can support developers in acquiring many quality attributes that are of interest to them. For example, the on-chain integration method, as an oracle feature, supports several quality attributes such as integrity, availability and transparency. In the Appendix, Table A2 demonstrates the full mapping.

4.2 Inference Engine

This section describes an inference engine that makes a secure and cost-effective optimal decision by computing the score of the feasible oracle platforms and excluding the infeasible ones; it ultimately

Table 3. Explanation of Attacks that Target Centralised (C) and/or Decentralised (DC) Oracle Types

Attack	Explanation	Oracle Type
Flash Loan Attack	An exploitation of a smart contract's security in which an attacker borrows a large sum of money without requiring collateral and then manipulates a crypto asset's price on one market before immediately reselling it on another.	C
Man-in-the-middle	A malicious actor intercepts the communication between the oracles and the contract and alters or falsifies the data.	C
Denial of Service	Attackers flood the data source with requests to slow it down or they prevent other users from using oracle's server by congesting the whole network.	C
SGX Attack	Executed by extracting attestation keys signed by Intel from SGX's private quoting enclave. The attacker can impersonate real Intel machines by signing arbitrary SGX attestation quotes.	C
Data Manipulation	When the data that the oracle collects have been tampered with at the data source.	C
Front Running	The transparency of data collected by oracles is used for personal gain.	C and DC
Malfunctions	Circumstances in which oracles offer skewed data despite the source being reliable and trustworthy.	C and DC
Sybil Attack	The Sybil node manipulates the decentralised oracle platforms to obtain the most votes by replicating their votes to gain a higher weight when compared to others.	DC
Mirroring	To assure the lowest cost of data gathering and the best possibility of selling the data to the client platform, the Sybil node gathers the data once and then distributes it to other nodes under its control.	DC
Freeloading	When a malicious entity duplicates data retrieved by another entity without making an effort to obtain the data itself.	DC
Spam Attack	An attack that drains the data feeder's balance with high gas fees.	DC
Sidechain Oracle Attack	An attacker can take control of oracle nodes in a sidechain to manipulate the data the oracles report and to change the outcome of smart contracts on the sidechain.	DC

recommends a ranked shortlist of feasible options. Steps six, seven and eight are involved with the inference engine.

4.2.1 Step Six (Features Selection). This step aims to specify the importance of each feature. Similar to Farshidi et al. [28], the MoSCoW [19] prioritisation technique is utilised to assign priority weights (W) to the oracle features. We employed this technique due to its simplicity, suitability for medium-to-large features and lack of complexity [38, 60]. WMoSCoW refers to four categories: wMust_have, wShould_have, wCould_have, wWon't_have where $\forall w \in WMoSCoW; 3 \geq w \geq 0$. We assigned a specific score to each category to be able to rank platforms based on numerical values. Thus, must-have and won't-have oracle features are given priority weights of 3 and 0, respectively, and act as hard constraints, while the features with should-have and could-have priorities are weighted 2 and 1, respectively, and act as soft constraints. A potential oracle platform must support all oracle features with must-have priorities, but not the features with won't-have priorities. For

example, if the decision-maker assigns must-have priority weight to the decentralisation feature, would mean that Provable Things and Town Crier, as centralised oracles, would be excluded from the ranked shortlist of potential platforms.

4.2.2 Step Seven (Score Calculation). The possible oracle alternatives are ranked by the Inference Engine based on their estimated scores using Equations (2) and (3). The **weight sum model (WSM)** [69] equation is applied to each supported oracle alternative to prioritise the oracle platforms based on the selected features. It is formulated as follows:

$$\sum_{i=1}^{n} W_i * Score_{PF(i)}, \tag{2}$$

where n denotes the number of features, W_i represents the weight assigned to the feature, and $Score_{PF(i)}$ represents the Boolean score of mapping the PF set. *Note:* If the resulting ranked list is large, the decision-maker can select certain threshold platforms and then apply Equation (3) to them.

4.2.3 Step Eight (Security Technical Debts Interest (STDI) Values and Monetary Cost Estimation). The STDI equation, proposed by Ahmadjee et al. [4], is adapted to estimate the security consequences of each oracle alternative. It is formulated as follows:

$$\text{STDI} = \text{CL} * \text{CAL} * \text{SR}, \tag{3}$$

where **`contract lifespan (CL)`** refers to the time, measured in days, between the contract's deployment and its last execution. The followings are the durations and corresponding weights: (i) Short-lived, 1–266 days, 0.17 score; (ii) Medium-lived, 267–533 days, 0.35 score; (iii) Long-lived, 34–800+ days, 0.5 scores.
`Contract activity level (CAL)` refers to the expected number of active users and the number of transactions that a contract is expected to deal with. A six-point scale is used to determine the activity level.
The `SR Value` is calculated as follows:

$$\text{SR} = \text{Likelihood} * \text{TI} * \text{BI}, \tag{4}$$

where `Likelihood` is the mean of the attack likelihood factors, based on the OWASP [48] risk rating methodology.
`Technical impact (TI)` is the mean of the TI factors, based on the OWASP [48] risk rating methodology.
`Business impact (BI)` is the mean of the BI factors, based on the OWASP [48] risk rating methodology.

Note: In the context of security, the debt interest value describes the negative consequences of exploiting a vulnerability. The longer the exploitable design flaw goes unresolved in the deployed contract, the higher the risk of interest accruing as a result of the flaw.

We propose a public framework that uses the aforementioned equations to help decision-makers quantify, estimate and prioritise oracle alternatives.

Finally, by adapting MOO to apply cost-value tradeoffs, the Inference Engine offers secure and cost-effective optimal solutions. The application of MOO techniques supports the selection of a set of secure and cost-effective solutions rather than one solution, as there is no unique option that can be the best fit with respect to security and low-cost objectives. The identified set of results is known as a **Pareto-optimal solution (POS)** [17]. Simple non-dominated sorting algorithms can be utilised to implement a vector ranking method that detects and indicates the elitism of each solution within a set of alternatives and returns a set of non-dominated solutions.

Table 4. Deployment and Transactional (Tx.) Cost of Integrating Each Oracle Platform with Ethereum Blockchain

Oracle Platforms	Deployment Cost	Tx Execution Cost
Provable	0.00161819 ETH	0.000299591 ETH
Chainlink	0.001632422 ETH	0.000306035 ETH
Witnet	0.003937 ETH	0.0075315 ETH
Tellor	0.002345111 ETH	0.00045238 ETH
BandChain	0.0052231 ETH	0.000400999 ETH
iExec	0.0072231 ETH	0.000372546 ETH

To clarify the application of the MOO technique and the related algorithms, we executed price feed smart contracts that fetch the Ethereum price from external sources and feed it to the contracts using several blockchain oracle platforms. This implementation allows us to recognise and compare the costs of deployment transactions and calling oracle transactions. The costs are calculated by adding the contract's deployment cost to the cost of a transaction that requests external data from an oracle. The cost required for deployment depends on the size of the smart contract and the part of the code that needs to be executed before the creation of the contract. The more complex the contract is, the higher the required cost. Interested readers can find a detailed explanation of the deployment cost in [4].

The cost of the oracle transaction mostly depends on the data validation mechanism's features and the number of data feeders participating in the validation process. Using a decentralised oracle requires two transactions to request data from an external source: The first transaction is responsible for sending tokens to the contract to be able to successfully implement the second transaction, which is requesting the price through the oracle. Table 4 shows the cost prices, in ETH, that are required to integrate each oracle alternative. For the sake of visualising and discussing the data, we have converted the costs to dollars after adding the cost of deployment to the cost of requesting data for each oracle alternative.

The values are the results of the STDI equation (Equation (3)) for each oracle platform. We estimated the STDI of each attack identified in step three, and the results are presented in Table 5. The likelihood and the impact factors were estimated based on the smart contract context and on the facts presented in the sources of knowledge that we used in previous steps. If the attacks have been successful, the corresponding likelihood estimation scores are higher than for unsuccessful attacks. Also, the quantified impact scores are higher if the attacks lead to catastrophic consequences. For instance, in November 2020, Cheese Bank was attacked for $3.3 million as a result of flash-loan attacks [52]. Other attacks such as mirroring [58] and freeloading [58] have not yet been successful. However, if these attacks do take place, the integrity of the entire system might be affected. We assume that the price feed contract is long-lived and that the contract is highly active. The explanations and scores for each factor are presented in detail in our public framework [2].

We added estimated values for the attacks that targeted centralised oracles and the values for the attacks that targeted decentralised oracles. For Chainlink, in addition to the total of STDI values for the decentralised oracle, we added the estimated value of SGX attacks because Chainlink supports SGX in its implementation. Sidechain oracle attacks mostly target BandChain as it uses the sidechain mechanism to provide cross-chain compatibility. Thus, for BandChain the estimated value of sidechain attacks was added to the total of STDI values for the decentralised oracle. The costs in dollars and STDI values of six blockchain oracle platforms are presented in Table 6.

Table 5. STDI Quantification of Attacks for Price Feeds Smart Contract

Type of Oracle	Attacks	Likelihood	TI	BI	CAL	CLS	STDI
Centralised	Data Manipulation	0.58125	0.475	0.45	5	0.5	1.344140625
	Flash Loan Attacks	0.46875	0.475	0.4875	5	0.5	1.127929688
	Man-in-the-middle	0.3375	0.65	0.43125	5	0.5	0.912304688
	Denial of Service	0.3375	0.625	0.45	5	0.5	0.90703125
	Front Running	0.46875	0.375	0.35625	5	0.5	0.856933594
	Malfunctions	0.525	0.225	0.4125	5	0.5	0.83671875
	SGX Attacks	0.39375	0.6	0.39375	5	0.5	0.978222656
Decentralised	Denial of Service	0.28125	0.475	0.13125	5	0.5	0.426269531
	Front Running	0.46875	0.425	0.35625	5	0.5	0.915527344
	Sybil Attacks	0.28125	0.425	0.4125	5	0.5	0.588867188
	Mirroring	0.28125	0.475	0.4125	5	0.5	0.588867188
	Freeloading	0.28125	0.475	0.4125	5	0.5	0.624023438
	Malfunctions	0.525	0.325	0.4125	5	0.5	0.96796875
	Spam attacks	0.4875	0.525	0.3	5	0.5	1.00546875
	Sidechain oracle attack	0.5438	0.7	0.3938	5	0.5	1.4868164

Table 6. Final Cost and Security Value of Oracle Platforms

Oracle Platforms	Cost	Value (STDI)
Provable	$8.37	5.941113281
Chainlink	$8.44	6.09521484375
Witnet	$45.87	5.1169921875
Tellor	$11.19	5.1169921875
BandChain	$22.49	6.603808594
iExec	$30.38	5.1169921875

To find the POSs, we need to select the solutions with minimum cost and minimum STDI value. If we compare Provable and Tellor, we notice that while Provable is better from the cost perspective, Tellor is better from the security angle. However, if we compare Tellor with Witnet and iExec, we notice that Tellor is equal to them in STDI values, but better than both of them in terms of cost. Tellor is better than BandChain in terms of both security and cost. Hence, Tellor dominates Witnet, BandChain and iExec. Similarly, Provable dominates Chainlink as it is better in terms of both security and cost. Therefore, Tellor and Provable form a non-dominated set, and they are POSs in the price feed smart contract example.

The inference engine finally offers a shortlist of feasible oracle platforms based on selected features, security assessment and monetary cost analysis. However, decision-makers can apply MOO analysis to perform further investigations, such as performance vs. cost, to find the best fitting blockchain platform for their application.

5 Application of the Model Using Two Case Studies

Two case studies of non-trivial scale were selected. The first describes the case of **dynamic legal agreements (DLAs)** [54], which are smart contracts that can adapt and respond to unpredictable

events, and which were used during the COVID-19 era (described in Section 5.1). The second case concerns **decentralised auction (DA)** applications [46] that capture the interactions between auctioneers and bidders (described in Section 5.2). Both cases leverage blockchain oracles, where the decision on selecting oracles is of paramount importance for dependable operations. In our analyses of these cases, we use the guidelines recommended by Runeson and Höst [55] and the framework proposed by Ebneyamini and Moghadam [23] to ensure clarity in the exposition and potential for replication and reproducibility, as described below:

(1) *Objectives of the case studies*: Our objective for presenting the case studies is to show how the blockchain oracle decision model can be applied in practice and to examine the applicability of the model in more than one case study. This allows us to ensure the usefulness of the proposed decision model for the oracle selection problem.
(2) *Reasons for the use of case studies*: We intentionally selected two smart contract projects as case studies as both will need to employ oracles that are suitable and effective fits for their applications. Even though the first case study employed an initial oracle platform, as the authors stated in the documentation, they are looking to integrate another oracle platform to improve the application. However, the second case study's developers have not decided which oracle platform to employ. Therefore, applying our decision model could assist in the selection of a feasible, secure and cost-effective oracle platform for both applications based on the features that the developers state in the documentation.
(3) *Methods of gathering data*: We searched for multiple sources such as documents, white papers, academic papers and repositories. These data sources allowed us to learn which oracle features are desirable and which specifications are required for each case study application. This is known as the third-degree method with regards to data collection [55], where available data are analysed and compiled independently without direct interaction with subjects. For the first case study, we were able to find several sources for data collection [54, 62, 63], while for the second, we were only able to find one source [46].
(4) *Data analysis*: Before we started the analysis process, we compiled the available data and prepared a template in an excel sheet where the oracle features could be filled in based on the extracted data. The analysis was done by one author and reviewed by the other authors. For both applications, we assume the blockchain is the best fitting technology because we are only interested in the selection and integration of oracle platforms. The oracle feature requirements were specified for both applications according to the MoSCoW priorities as shown in Table 7. Then, the inference engine recommended feasible solutions based on feature priorities and POSs. Finally, we compared our decision model outcomes with outcomes using *ad hoc* methods.

The remainder of this section describes the applications in both case studies and discusses the outcome of applying the decision model.

5.1 Application One: DLA

A DLA [54] is a smart contract that can adapt and respond to an unpredictable event with severe consequences, such as the COVID-19 pandemic. The contract needs to employ an oracle to fetch the required data regarding the event, such as the rates of virus infection and transmission, from external sources and feed it to the contract. Therefore, the agreement can be adapted based on the provided data. An example might include a service level agreement relying on the supply of goods being limited due to the closure of a production line. This application is implemented using the Provable oracle platform. However, as mentioned above, the developers are looking to integrate another oracle platform to enhance the application. Thus, applying our decision model assists in

Table 7. Entire List of Oracle Features of the Two Case Studies

MoSCoW	Dynamic Legal Agreements	Decentralised Auctions
Must-have	Multiple sources	Multiple sources
	Consensus-based solution	Consensus-based solution
	Staking and reputation	Staking and reputation
Should-have	Fully decentralised oracle	Fully decentralised oracle
	On-chain	On-chain
	Hybrid	Hybrid
	Median aggregation mechanism	Median aggregation mechanism
	Statistical measure	Rewarded incentive scheme
	Staking dispute resolution	Staking dispute resolution
	Slashing punishment	Slashing punishment
	Data correction	Data reversion
	Rewarded incentive scheme	Reputation loss
	PoW, PoCo, random and pseudo-random	Existence of a native token
	Existence of a native token	
Could-have	Single data source	Single data source
	Semi-decentralised oracle	Semi-decentralised oracle
	Trusted execution environment	Trusted execution environment
	Symmetric and asymmetric encryption	Symmetric and asymmetric encryption
	Mode aggregation mechanism	PoW, PoCo, random and pseudo-random
	Mean aggregation mechanism	Mode aggregation mechanism
	Voting aggregation mechanism	Mean aggregation mechanism
	Reputation loss	Voting aggregation mechanism
	Banding	Statistical aggregation mechanism
	Statistical measure for dispute resolution	Banding
	Data reversion	Data correction
	Using of a native token	Statistical measure for dispute resolution
	Using Ether	Using of a native token
		Using Ether
Won't-have	Centralised oracle	Centralised oracle
	Trusted third-party	Trusted third-party
	Off-chain	Off-chain

the selection of a feasible, secure and cost-effective oracle platform for the DLA contract based on the features that the authors state in the documentation.

The DLA application was developed by the CTO-in-Residence at the UCL School of Management and a business analyst, who collaboratively contributed to providing contractual certainty in an uncertain time. They wrote a discussion paper [54] that was published by the UCL Centre for Blockchain Technologies and participated in a hackathon [62, 63] where all their code was made public in online repositories [63].

The following are some of the expected application functionalities and related oracle feature requirements extracted from the available sources. The MoSCoW technique is utilised to assign priority weights (W) to the desirable oracle features.

(1) Since the developers of the DLA application wanted to update the application and employ multiple entities to provide data integrity, trustlessness and availability of the fetched data

needs to be ensured. A *fully decentralised oracle* is a suitable type, and it is considered a should-have feature. A *semi-decentralised oracle* is prioritised as could-have, while won't-have priority weight is assigned to a *centralised oracle*.

(2) The developers aim to get the most accurate data for the contracts to use. Integrity, therefore, must be ensured. This type of application requires retrieved data to be trustless and available as the data need to be sourced from multiple trusted sources. Therefore, *multiple sources* need to be leveraged and prioritised as a must-have. A *single source* is considered a could-have feature.

(3) The developers' intention is to avoid centralisation in the updated application to better meet qualities like security, availability and fault tolerance. Hence, a *trusted third-party* is considered a won't-have feature as it introduces centralisation to the application. A *consensus-based solution* is a must-have feature as it is more applicable to decentralised applications. A *trusted execution environment* can be utilised if the data need to be processed in a protected environment. This feature is therefore considered a could-have.

(4) The developers aim to provide a means of transparency in the application. Thus, *on-chain* integration or *hybrid* integration are the suitable integration methods and are prioritised as should-have, while off-chain integration is prioritised as won't-have.

(5) The application does not have strict requirements for data confidentiality and sensitivity. Therefore, both *symmetric* and *asymmetric encryption* methods are prioritised as could-have oracle features.

(6) Since the integrity and correction of the data are important in DLA applications, *staking* and *reputation* can be leveraged as selection methods for data feeders, and they are both considered must-have features. The remaining features, such as *PoW, PoCo, random and pseudo-random*, can be considered should-have features as they will increase the integrity of the data even if they lead to transaction latency because the speed of transactions in the DLA application is not a crucial issue.

(7) The *median aggregation mechanism* is a should-have feature in the DLA application as most of the required data is numerical, such as the number of infections. However, some non-numerical data, such as time periods for restrictions to be started, might also be required, and *mode* is more suitable for use on non-numerical data. Therefore, applying the *statistical approaches* and letting the data requesters select a suitable method based on the nature of the external data is more appropriate. Thus, *statistical approaches* are also a should-have feature, and the rest of the aggregation methods are considered could-have features.

(8) To ensure the quality of the result, *dispute resolution* based on *staking* is a more feasible solution as it enforces the feeders to behave honestly. Otherwise, they could suffer economic loss by having their stake slashed. Therefore, *staking* and *slashing* are should-have features as dispute resolution and punishment methods, respectively. Additionally, a *statistical approach* can be employed in combination with staking by slashing the collateral of the feeder if the data deviate from the median by some threshold. Data feeders can be punished not only by losing their collateral tokens but also by reducing their reputation rank on the reputation registry. The loss of tokens is an immediate penalty, but the loss of reputation may influence future revenue. As a result, the *statistical approach* and *reputation loss* are could-have features for dispute resolution and punishment, respectively. *Banding* is considered a could-have feature.

(9) Since the speed of the transaction is not crucial in the DLA application and the integrity of the data is important, *correction* is a should-have feature for dispute data. Data correction demands the submission of new data, followed by a decision taken on all the options. A could-have priority is assigned to the *revision* feature.

(10) A *reward incentive scheme* is a should-have feature as all decentralised oracle platforms use rewards as an incentive to encourage data feeders to be honest. Similarly, the *existence of a native token* is a should-have feature as each decentralised oracle has its own token.
(11) The DLA application has no strict requirements regarding which token is used when requesting external data. Thus, a could-have priority weight is assigned to both features (required and not required).

5.2 Application Two: DAs

This DA application [46] is a blockchain-based solution that uses an Ethereum smart contract, a decentralised storage system and oracles to capture the interactions between auctioneers and bidders. The oracle acts as an external timer that indicates the bidding start and end times of the contract. Even though the developers of the DA application have thoroughly discussed and analysed it, in their paper, they have not specified which oracle platform will be integrated into their application. Our blockchain oracle selection decision model can overcome guesses and *ad hoc* practices; should it be used in similar contexts, it can provide systematic means for informing the selection of more secure and cost-effective oracle alternatives that can be integrated into the Ethereum DA application.

The auction application project was proposed by five researchers—three professors and two research associates—whose work was supported by the Research Center for Digital Supply Chain and Operations Management at Khalifa University. Although we were only able to find their published academic paper [46] regarding the auction application, we were able to extract the main oracle requirements from the paper. The criteria and the required oracle features that were stated by the developers of the DA application in their paper are listed below. The MoSCoW technique is utilised to assign priority weights (W) to the oracle features.

(1) The main objective of the DA developers was to avoid placing trust in a single source to avert having a single point of failure. Applying a centralised oracle would reintroduce this issue to a decentralised environment. Therefore, a *fully decentralised* oracle is a suitable type, and it is considered a should-have feature, while a *semi-decentralised* oracle and *centralised* are considered could-have and won't-have features, respectively.
(2) The DA application also tries to avoid using a *single source* for its time data to increase availability, integrity and trustlessness. Thus, *multiple sources* need to be leveraged and prioritised as a must-have. A *single source* is considered a could-have feature.
(3) Since third-party intermediaries need to be eliminated, a decentralised environment needs to be established in the DA application. Therefore, a *trusted third-party* is considered a won't-have feature and a *consensus-based* solution is a must-have feature as it is more appropriate for decentralised applications. A *trusted execution environment* is a could-have feature as it could be useful if the data need to be processed in a protected environment.
(4) Enhancing transparency and tractability of online auctions are the main objectives of developing the DA application. Thus, *on-chain* integration or *hybrid* integration are the suitable integration methods and are prioritised as should-haves, while *off-chain* integration is prioritised as won't-have.
(5) The application does not have strict requirements for data confidentiality and sensitivity. Therefore, both *symmetric* and *asymmetric encryption* methods are prioritised as could-have oracle features.
(6) Since the developers of DA intend to address the core security concerns related to data integrity in online auctions, both *staking* and *reputation* are considered must-have features that can be leveraged as selection methods for data feeders. Although applying the remaining

Table 8. Inference Engine Outcomes Based on Features' Priorities and POS

Oracle Platforms	WSM (DLA)	POS (DLA)	WSM (DA)	POS (DA)
Tellor	**33**	Chainlink Tellor	**30**	Provable Tellor
iExec	**32**		**31**	
BandChain	**31**		**30**	
Chainlink	29		29	
Paralink	27		28	
Witnet	29		28	
Town-Crier	8		8	
Provable	4		4	

The bold numbers represent the top three Oracle platforms, ranked based on the calculated WSM (Weight Sum Model).

features, such as *PoW, PoCo, random and pseudo-random*, increases the integrity of retrieved data, it also increases transaction latency which the DA application requires to be low. Thus, a could-have priority is assigned to these features.

(7) In the DA application, time is calculated using the Unix epoch time in Solidity, which considers time as a number. Thus, the *median* aggregation mechanism is a should-have feature in the DA application as it is more suitable for numerical values. The rest of the mechanisms are considered could-have features as they might be applied if time were to be treated as a string.

(8) The DA developers refer to the necessity of employing an oracle platform that rewards truth-reporting nodes while punishing malicious nodes in the network. Therefore, a *rewarded incentive scheme* is a should-have feature. *Slashing* and *reputation loss* punishment methods are also should-have features, while *banding* is a could-have feature.

(9) Since transaction speeds are crucial in the DA application, *data reversion* is a should-have feature for dispute data as it is faster and less complex than *data correction*, which is considered a could-have feature.

(10) To ensure the quality of the result, *dispute resolution* based on *staking* is a more feasible solution as it enforces the feeders to behave honestly. Otherwise, they could suffer economic loss by having their stake slashed. Therefore, *staking* is a should-have feature for dispute resolution. Additionally, a *statistical approach* can be employed in combination with staking by slashing the collateral of the feeder if the data deviate from the median by some threshold. Thus, the statistical approach is a could-have feature for dispute resolution.

(11) The *existence of a native token* is a should-have feature, as each decentralised oracle has its own token.

(12) The DLA application has no strict requirements regarding which token should be used when requesting external data. Thus, a could-have priority weight is assigned to both features (required and not required).

5.3 Results and Analysis

The blockchain oracle features were specified according to the MoSCoW priorities. Then, the inference engine of the oracle decision support model identified optimal solutions for each application. Table 8 shows the oracle alternatives along with their calculated scores based on WSM. The table also indicates the POSs for each application. Although the estimation of STDI factors was based on the attack information extracted in Step 4, certain factors, including some SR factors, CL and CAL, might need to be updated based on the application context. STDI calculations for each application

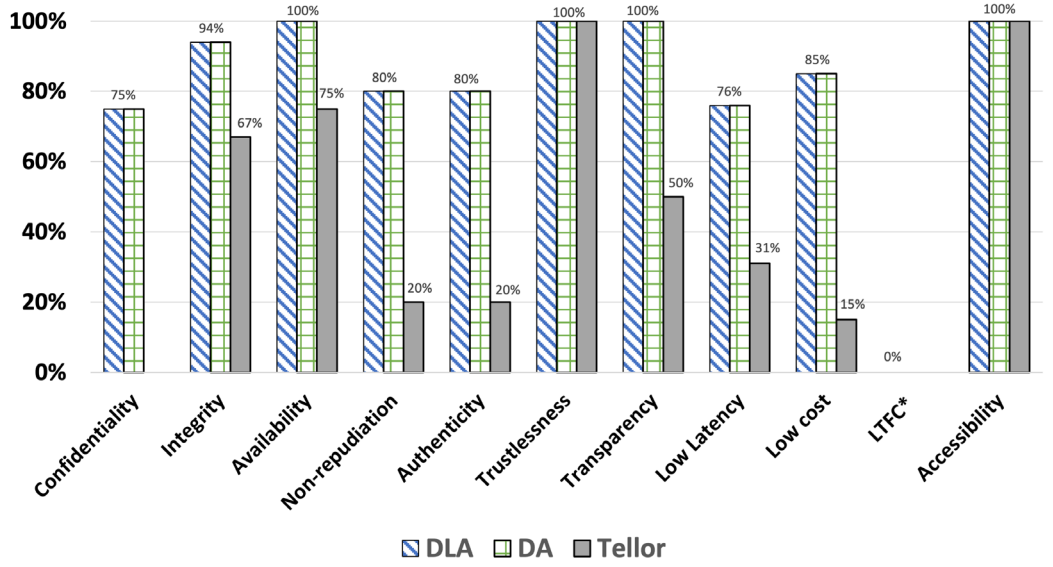

Fig. 3. The percentages of the Q that are desired for each application and the percentages that Tellor provides.

are presented in detail in our public framework [2]. The monetary cost of integration for each oracle is demonstrated in Table 5.

Based on the oracle features that we extracted from the DLA documentation, the oracle decision support model assigned the highest score to the Tellor oracle platform. Tellor supports all the must-have, and most of the should-have and could-have, oracle features. Moreover, Tellor is one of the POSs as it is more secure and provides a better cost-benefit than any of the other decentralised solutions. The second most feasible oracle platform is iExec, which has higher scores than the BandChain, Chainlink and Paralink platforms. The Witnet platform is excluded as it does not support one of the must-have features, which is the staking method for data feeder selection. Provable and Town Crier are also excluded as they fail to support more than one of the must-have features, such as decentralisation and consensus data validation methods.

Regarding the DAs application, the oracle decision support model also assigned the highest scores to the iExec, Tellor and BandChain oracle platforms. They support all the must-have, and most of the should-have and could-have, oracle features. However, Tellor is better than both BandChain and iExec as it provides a better cost-benefit than all other decentralised solutions. The fourth most feasible oracle platform is Chainlink, which has higher scores than Paralink platforms, which ranked as the fifth most feasible solution. The remaining three platforms, Witnet, Provable and Town Crier, are excluded as they do not support one or more of the must-have features.

The inference engine concludes that Tellor is a feasible oracle platform for both applications, which means that this platform at least supports all oracle features with must-have priority and does not support the features with won't-have priority. Additionally, it provides better security and cost-benefit than the other platforms. The developers of both applications had not considered Tellor as a potential feasible oracle platform for their applications. DLA employed Provable in the current version and is considering employing Chainlink for the updated version as stated in the documentation. Although the DA developers have not integrated any oracle yet, they considered Provable, Chainlink and Witnet as candidate platforms to be implemented in the DA application.

Figure 3 shows the percentages of the quality attributes based on the selected oracle features that need to be implemented in each application. Additionally, the figure demonstrates the percentages

Table 9. Sensitivity Analysis of WSM Results of DLA Application

Features	Main Scenario			Scenario 1			Scenario 2		
Data Feeders Selection Method	**Weight**	**Oracle**	**Score**	**Weight**	**Oracle**	**Score**	**Weight**	**Oracle**	**Score**
Staking	**3**	Tellor	33	**2**	Tellor	30	**3**	Tellor	35
Reputation	**3**	iExec	32	**2**	iExec	28	**3**	iExec	34
PoW	**2**	BandChain	31	**1**	BandChain	28	**3**	BandChain	32
PoCo	**2**	Chainlink	29	**1**	Chainlink	27	**3**	Witnet	31
Random	**2**	Witnet	29	**1**	Witnet	26	**3**	Chainlink	29
Pseudo-random	**2**	Provable	4	**1**	Provable	4	**3**	Provable	4

The bold number indicates the change in the technical impact score compared to the main scenario.

of the quality attributes that the Tellor platform provides. Availability and integrity are the main objectives that both applications need to meet. Tellor provides high availability and integrity with 75% and 67%, respectively. Transparency is also a primary feature required in both applications and Tellor provides a reasonable percentage of 50%. Tellor affords a high percentage for trustlessness, which is one of the main aspects that needs to be provided in the DA application. Confidentiality is not provided by Tellor. However, both applications do not have strict requirements for data confidentiality and sensitivity. Providing a LC solution is the objective of the DA application, and based on our experiments, Tellor provides lower costs than most of the other decentralised alternatives. Moreover, Tellor is resistant to MITM attacks and data manipulation, which are the main concerns stated in the DA documentation. Even though Tellor is better than other platforms in terms of security, cost and most of the other attributes, Tellor is slow as it provides an LL of 31% because the dispute resolution step may take a long time to resolve. Therefore, developers might need to make a further tradeoff analysis to compare and investigate the applicability of other platforms. The analysis can be done using our decision model framework.

6 Evaluation

In this section, we evaluate our decision model and appraise its reliability and usefulness. We report on the sensitivity analysis, discuss the added value of our method when compared to *ad hoc* selection practices, and explore how we go beyond existing work in providing an explicit oracle selection method.

6.1 Sensitivity Analysis

Sensitivity analysis [11] is vital for evaluating our decision model's quality and stability when confronted with uncertainties and input value changes. It helps gauge the resilience of our estimations and rankings against altering values, which might result from input uncertainty or the need to explore the design space. In our study, we assessed the sensitivity of our blockchain oracle decision model by modifying weight values in the WSM equation to observe the impact on oracle alternative rankings. Additionally, we adjusted some factors in the STDI equation to test how results varied with different attack severity levels.

The key to sensitivity analysis lies in identifying the most influential assumptions affecting the output, or which input variables exert the strongest impact on the target variable. We focused on modifying weights in oracle categories with the highest number of features, notably the *Data Feeders Selection Method* in our model. The resulting scores were re-calculated according to the updated group weights. Our sensitivity analysis was applied to the DLA application example, with the results presented in Table 9.

Table 10. Sensitivity Analysis of STDI Results

Attack	TI Factors								Final Result
Flash Loan Attacks	Loss of Confidentiality	Score	Loss of Integrity	Score	Loss of Availability	Score	Loss of Accountability	Score	STDI
Scenario 1 (main)	Minimal non-sensitive data disclosed	2	All data totally corrupt	9	Minimal secondary services interrupted	1	Possibly traceable	7	1.1279297
Scenario 2	Minimal critical data disclosed	**6**	All data totally corrupt	9	Minimal secondary services interrupted	1	Possibly traceable	7	1.12451172
Scenario 3	Minimal non-sensitive data disclosed	2	Extensive seriously corrupt data	**7**	Minimal secondary services interrupted	1	Possibly traceable	7	1.0693359
Scenario 4	Minimal non-sensitive data disclosed	2	All data totally corrupt	9	Minimal primary services interrupted	**5**	Possibly traceable	7	1.12451172
Scenario 5	Minimal critical data disclosed	2	All data totally corrupt	9	Minimal secondary services interrupted	1	Completely anonymous	**9**	1.1865234
Scenario 6	Minimal critical data disclosed	**6**	Extensive seriously corrupt data	**7**	Minimal primary services interrupted	**5**	Completely anonymous	**9**	1.3623046875

The bold text highlights the top Oracle platforms in each category.

Sensitivity analysis is applied here by changing the priority weighting according to the MoSCoW [19] scheme using alternative scenarios. We initially explored two scenarios to understand the impact of adjusting the weight parameter:

In Scenario 1, we decreased the weight parameter by one. This change meant that features categorised as 'should-haves' (2) in the primary scenario were treated as 'could-haves' (1). In Scenario 2, we increased the weight parameter by one. Consequently, 'should-have' features (2) in the main scenario were elevated to 'must-have' status (3).

We observed that these adjustments caused variations in the final scores for each alternative, resulting in minor changes to their rankings. However, Tellor maintained its position as the top-ranking platform in comparison to the other alternatives. It's worth noting that the model exhibits sensitivity when dealing with hard constraints (must-have and won't-have weights). These constraints influence the inclusion or exclusion of platforms in the evaluation. For instance, assigning a 'should-have' weight instead of a 'must-have' weight to the staking feature led to the inclusion of Witnet among the suggested platforms. This was despite the fact that Witnet should have been excluded due to its lack of support for one of the 'must-have' features.

A **one-factor-at-a-time (OFAT)** [61] approach has been applied to evaluate the sensitivity of STDI estimation outputs against the assigned set of factor values. Five alternative scenarios (Scenario 2 to Scenario 6) were constructed, as shown in Table 10, to apply the analysis. These scenarios are effective in revealing the sensitivity of the STDI estimations. In particular, we modified the values of the TI factors because they have a higher impact on the final estimation score compared with the two other factors, namely the likelihood and the BI. We slightly changed the range of one technical impact factor each time by increasing or decreasing one of the values. For example, we changed the 'loss of availability' technical impact factor for flash loan attacks in Scenario 4, from 1 (minimal secondary services interrupted) to 5 (minimal primary services interrupted), which is the subsequent score values based on the OWASP [48] risk rating methodology. Then the final STDI result was re-calculated based on the updated score. This analysis indicated that the STDI estimation is stable, as the final estimations of the attacks' severity range ($1.00 \leq \text{STDI} < 1.5$) did not change. Additionally, we simultaneously modified the values of the TI factors in Scenario 6 to observe the changes in the final estimation values. As with OFAT, the final estimations of the attacks' severity range remained unchanged.

Table 11. Our Work in Comparison to Previous Works

Oracle Features	[44]	[5]	[40]	[25]	[51]	[67]	[33]	**This Work**
Type of Data Feeder	×	✓	×	✓	✓	✓	✓	✓
Type of Data Source	✓	×	✓	✓	✓	×	✓	✓
Data Validation Mechanism	✓	✓	✓	×	✓	×	✓	✓
Integration Methods	✓	✓	×	×	×	×	×	✓
Encryption Method	✓	✓	×	×	×	×	×	✓
Aggregation Mechanism	×	×	×	✓	✓	×	×	✓
Dispute Resolution	×	×	×	✓	✓	×	×	✓
Incorrect data	×	×	×	✓	×	×	×	✓
Incentive scheme	×	×	×	×	×	×	✓	✓
Punishment methods	×	×	×	✓	×	×	×	✓
Native Token (NT)	×	×	×	×	×	×	×	✓
Using NT	×	×	×	×	×	×	×	✓
Data Feeders Selection Method								
Staking	×	×	×	✓	✓	×	✓	✓
Reputation	×	×	×	✓	✓	×	✓	✓
PoW	×	×	×	✓	×	×	×	✓
PoCo	×	×	×	×	×	×	×	✓
Random	×	×	×	×	×	×	✓	✓
Pseudo-random	×	×	×	×	×	×	×	✓

6.2 Comparison with Previous Work

This section considers the academic papers we selected in the knowledge acquisition and organisation phase. We also applied a snowball method to ensure that all studies related to our work are included. This section's objective is to compare our decision model's novelty and effectiveness against existing work related to blockchain oracles. We compared our work to the academic literature in terms of the number of oracle features discussed in *step 2* and quality aspects covered in *step 3*, as well as the type of methods that have been used.

Table 11 illustrates the oracle features covered by each study. As shown in the table, the majority of studies agreed concerning certain essential features such as the *type of data feeder*, *type of data source* and *data validation mechanism*. Some features, such as *integration methods* and *encryption methods*, are only discussed in a few studies. Only one study [25] analysed *aggregation mechanisms*, *dispute resolution*, *incorrect data*, *punishment methods* and several methods of *data feeder selection*. Our model has compiled additional features that had not been covered by any other work, including *incentive schemes* and the *existence of and use of native tokens*. These features allow decision-makers to anticipate the monetary cost of oracle integration. Additionally, only our model involves *random* and *pseudo-random* techniques as features concerning data feeder selection methods. Considering these features at the decision-making stage assists in recognising the extent to which the chosen oracle is secure [7]. Our model considers more features than other works that have discussed blockchain oracles. Even though there is no agreement on the number of features that should be involved in the decision-making process, enlarging the set of features can provide more meaningful comparisons between oracle alternatives, which may lead to the selection of a more optimal solution. Although our model is more comprehensive regarding the number of features, it gives

decision-makers a degree of flexibility when it comes to customising the selection so that analysts can focus on the key features relevant to their situation.

Recently, Farshidi [26] introduced a theoretical framework [28] to assist software engineers with a set of MCDM problems in software production. The framework provides a guideline for software engineers to systematically capture knowledge from different knowledge sources to build decision models for MCDM problems in software production. Knowledge has to be collected and organised when it is needed to be employed. The framework and a Decision Support System [27] were introduced in their previous studies for building MCDM decision models in software production [9, 29, 30].

Unlike previous works, we have linked each oracle feature to corresponding quality attributes, as illustrated in Table 2 and Appendix A. Mammadzada et al. [44] discussed the confidentiality and integrity provided by some oracle features. Al-Breiki et al. [5] compared the integrity, confidentiality, authenticity and accessibility of the oracle solutions. Lo et al. [40] investigated the reliability of seven oracles in terms of their probability of failure. Xu et al. [67] considered availability and transparency when discussing oracle features. However, none of these studies discussed the security implications of the analysed oracle platforms. An examination of current approaches addressing the challenges posed by Blockchain oracles is conducted by Hassan et al. [33], with a specific focus on the security services they provide within the security triad (confidentiality, integrity, availability), along with an exploration of their constraints.

Some studies explored and analysed the potential security vulnerabilities and the types of attacks that target blockchain oracles. Eskandari et al. [25] analysed oracle design options, explored potential system vulnerabilities, discussed cyberattacks and showed some attack mitigation measures. They also mentioned examples of some real attacks that had severe consequences. Pasdar et al. [50] analysed Sybil attacks that mostly target decentralised oracles.

Analysing the concept of the oracle problem and illustrating some of the issues with blockchain oracles has been done by Hassan et al. [33]. The article explains the malicious behaviour of attackers and mentions a couple of threat sources, such as third parties and governments or corporations, which may deliberately modify the data through oracles or shut down the functionality of an oracle, akin to carrying out a DDoS attack. Furthermore, Wang et al. [64] and Aspembitova and Bentley [6] have contributed to the field by examining attacks and threats within the realm of DeFi, with a particular focus on price manipulation tactics.

Even though the studies discussed some attacks and their sources that targeted oracles, they failed to quantify the consequences of the SRs. In our model, we classify and assess the possible attacks based on their likelihood, possible impacts and type of smart contract. Moreover, to the best of our knowledge, we are the only study that has discussed and compared the monetary cost of each oracle alternative.

To the best of our knowledge, there is a dearth of academic studies that provide a systematic approach to comparison and evaluation of oracle alternatives and which assist smart contract developers in selecting feasible solutions for their applications. Mammadzada et al. [44] conducted a systematic literature review to formulate an oracle framework that defines what blockchain oracles are, and how they relate to blockchain applications. Al-Breiki et al. [5] investigated the trustworthiness of the multiple oracles and their system design, benefits and drawbacks. A fault tree analysis was utilised by Lo et al. [40] to study oracle systems' dependability and to analyse their designs. Eskandari et al. [25] conducted a systematisation of knowledge study to present a classification of existing oracle implementations. Pasdar et al. [51] provided a comprehensive review and categorised oracles into two main groups: voting-based and reputation-based. Xu et al. [67] briefly discussed oracle problems by providing a simple decision model for connecting blockchain systems with the external world.

Table 12. Comparison of Our Model and *Ad Hoc* Methods

Our model (all steps)		**Cost Only**	**STDI Only**	**WSM Only**
WSM	**POS**			
iExec	Provable	Provable	Witnet	iExec
Tellor	Chainlink	Chainlink	Tellor	Tellor
BandChain	Tellor	Tellor	iExec	BandChain
Chainlink	BandChain	BandChain	Provable	Chainlink
Witnet	iExec	iExec	Chainlink	Witnet
Provable	Witnet	Witnet	BandChain	Provable

Even though the methods applied in previous works might implicitly assist in the oracle selection process, we are not aware of any systematic method that provides a step-by-step approach when it comes to the selection of a suitable oracle platform. In the following Section 6.3 we have provided analysis of the value added by our systematic model when compared to prevalent *ad hoc* methods of selection.

6.3 Comparison with *Ad Hoc* Methods

Our comparison is based on the fact that our systematic model obliges decision-makers to consider an application's required features, cost and security attributes. However, decision-makers can opt not to examine any of these aspects when picking an oracle platform on an *ad hoc* basis. Indeed, ignoring any of these aspects can lead to a different, potentially sub-optimal, outcome. Table 12 shows the results of our comparison.

By applying all the steps of our model for the DA application, we found that iExec had the highest WSM scores followed by Tellor. However, Tellor was better in terms of cost. If the cost attribute is not considered, iExec might be employed despite being more expensive than Tellor. In contrast, if cost is the only factor in the decision process, Provable would be the best choice, followed by Chainlink. However, neither platform is optimal for DA applications, as Provable has several unsuitable features, and Chainlink has a high STDI value. Finally, if the security attribute is the only consideration, Witnet, Tellor and iExec would be the best options. However, not considering the required features of the application and the cost of employing the platform might lead to the selection of a sub-optimal platform for an application such as Witnet. That platform is unsuitable for DA applications, as it is the most expensive platform and does not provide features that must be applied in DA applications.

7 Discussion

The existence of multiple blockchain oracle platforms with varying features and criteria makes it difficult for decision-makers to select the best platform for their applications. Based on expert interviews, blockchain oracle platforms are currently chosen in an *ad hoc* manner. One of the experts stated, '*Developers often choose a blockchain oracle platform based on its reputation or rely on the wisdom of the crowd'*. Another expert expressed, '*Developers frequently treat blockchain oracles as black boxes and lack a comprehensive understanding of their underlying mechanisms'*. Thus, our decision model comes to overcome *ad hoc* practices and assist smart contract developers in the systematic selection of feasible, more secure and cost-effective oracle solutions that the developers may not be able to discern without the support of the proposed model. The importance of our decision model was also emphasised by blockchain oracle co-founders, one of them stated '*Existing*

of such a model facilitating transparent oracle selections by making oracle alternatives and their associated features and qualities more explicit'. The model can be applied at the architectural design stage, where architects and blockchain software engineers have to integrate oracle solutions into the software being designed, by probing for features that best meet the quality attributes of interest for the said application. The systematic steps that we provide as part of our model serve this objective.

One of the distinctive features of our model is that it provides cost analysis, covering the deployment and execution of the oracles under investigation. These are particularly important factors to consider when integrating oracles with Ethereum smart contracts because the cost can differ among platforms. For example, using the Provable platform comes with a different cost compared to using Chainlink. The observation is also evidenced by previous studies, which indicate that different oracle features can lead to different costs [5, 34]. Additionally, one study [50] emphasised the need for further research that investigates the operating costs of blockchain oracles. Unlike previous studies, our study implements price feed contracts to compare and analyse the exact costs of each oracle.

Another distinctive feature of our model is the provision of a security technical debt assessment for the selection. Our technical debt analysis considers inherent issues in the oracle that can make the solution more susceptible to attacks that target blockchain oracle platforms. The evolution of oracles has led to an increase in attacks which demands a greater focus on security analysis and assessment [13, 25]. Using our model, developers can thus make informed decisions before integrating oracle platforms into the smart contract. Applying a security debt assessment step helps to reduce the introduction of unintentional debt caused by unawareness of external security issues related to oracle platforms, while increasing the visibility of such issues and helping manage the debt more strategically. Furthermore, smart contract developers can avoid debt that might accumulate as a result of incorrect oracle selection decisions that would lead to re-selection, re-development, and redeployment of the contract.

Selecting an oracle with a scope mismatch to the application's requirements can leave systems vulnerable to significant attacks [13]. For instance, in 2020, Curve Finance faced intense scrutiny due to numerous flash loan attacks totalling over one hundred million dollars. The attackers exploited the widespread adoption of Curve Finance **liquidity providers (LP's)** price feeds by various DeFi platforms [57]. These feeds, linked to LP pools, exhibit significant volatility to swiftly adjust to pool size and usage. This volatility was exploited by perpetrators to execute flash loan attacks, finding them both straightforward and lucrative. Despite Curve Finance's data being publicly accessible, their LP price feeds were not originally intended for use as collateral pricing sources. Surprisingly, the Curve Finance team acknowledged in various publications their unawareness of other DeFi projects leveraging their LP price feed, prompting them to recommend alternative specialised oracle services.

Our decision model indicates that there is no unique optimal solution that can be the best fit for all quality attributes. This should come as no surprise. The model suggests more than one feasible oracle platform based on the required features selected by the developer, the security assessment, and the cost analysis. Mapping quality attributes to each oracle feature and each oracle platform permits the identification of quality attribute tradeoffs related to the chosen features and suggested oracle solutions. This assists in choosing among the oracle alternatives while bearing in mind the desired quality attributes.

Systematic and fine-grained analysis like our decision support model can uncover issues that might be ignored when taking an *ad hoc* approach or even when consulting with experts. Without the use of approaches such as our model, a sub-optimal design decision might be made. Furthermore, choosing a more optimal solution reduces the overhead of re-designing and updating the smart contract, which therefore reduces the overall cost of the contract's development.

The application of our model can benefit both uninformed and knowledgeable decision-makers. The model provides information on oracle platforms to assist inexperienced decision-makers while also providing experienced ones with a systematic model. An investigation into our model by industry would likely contribute towards a fuller understanding of the costs and benefits that its introduction at the decision-making stage would provide. According to the interview and survey by Zou et al. [72], there is a lack of standardised knowledge and useful guidance for developing secure smart contracts, and our model aims to fill that gap. Practitioners also refer to the need to define criteria and features for selecting the most appropriate smart contract components [53].

7.1 Threats to Validity

Based on [65] and [55], we have considered the following potential threats to validity: internal validity, conclusion validity, construct validity, external validity and reliability.

Internal Validity. A potential threat to internal validity is the completeness of the knowledge extraction steps. We mitigated this risk by conducting a thorough inspection of multiple sources, including the online documentation for each platform, industry white papers, academic papers and blogs. The landscape of oracle features and cyberattacks is continuously evolving and completeness cannot be guaranteed. Nevertheless, our work provides comprehensive and detailed coverage that goes beyond existing works, as discussed in Section 6.2. New attacks can always emerge and attackers may exploit hidden security flaws in oracles; the visibility of these attacks can then become noticeable over time. However, our decision model is adaptable and flexible to evolve and cope with new additions and changes. Researchers can add more oracle platforms, oracle features and/or possible oracle attacks to the decision model by systematically following the five steps of knowledge acquisition and organisation that we presented.

Conclusion Validity. A potential threat to conclusion validity is related to the need to involve decision-makers for blockchain oracle applications in the examination of the applicability of the decision model. However, we mitigated this threat by using public and well-documented applications for the decision model application.

Construct Validity. There is a construct threat regarding the possibility of inaccuracy when assigning scores to the impact and likelihood factors. However, the sensitivity analysis we conducted demonstrates the stability of the STDI estimation approach. Furthermore, our framework is public [2] and allows decision-makers to change the scores based on their context. Another threat might be related to employing the MoSCoW prioritisation technique rather than another technique. Based on Farshidi et al. [28, 29], who evaluated their decision models using multiple experts and according to the comparative analysis conducted by [38], and [60], the MoSCoW technique is simple, suitable for large-medium features, easy to use, and imbues users with confidence when used. For these reasons, we chose to employ the MoSCoW technique in our model. Nevertheless, researchers can use other types of prioritisation techniques to define their feature requirements. Regarding the data collection in terms of the case studies, there is a potential threat related to misinterpretation of the oracle features required by each application. To mitigate this, two authors independently reviewed the extracted features. Whenever there was a disagreement, all authors discussed the matter until a conclusion was reached.

External Validity. There is an external threat related to the applicability of our decision model to other types of blockchain oracle platforms, i.e., not just Ethereum oracle platforms. Ethereum is, however, the most used platform for developing decentralised applications. Nevertheless, the decision model presented here can be updated to involve more types of oracle platforms without changing the main steps. Moreover, we applied the model in two domains, insurance and auction application domains, to prove that our decision model can be generalised to other contexts.

Reliability. To strengthen the reliability of our study, all the data sources and information related to knowledge acquisition and mapping have been made available. We have also been open and explicit about the calculation process used in the inference engine. Additionally, the data sources for the case study applications have been given, and the templates for the extracted features have been made available to the public [2]. This enables easy replication of the case study analysis with a high likelihood of achieving the same outcomes.

8 Conclusion and Future Developments

Automation Support. In our ongoing work, we seek to build decision support systems that are fully automated to facilitate faster and more efficient analysis. Automated support empowers smart contract software engineers to assess the potential implications of adopting alternative oracles during contract design. Furthermore, our investigation encompasses an analysis of the timeliness and sensitivity associated with data transmission. Concurrently, our ongoing efforts explore the utilisation of machine learning and search-based software engineering techniques. These aim to autonomously predict and select features critical for a given context, optimising the selection process. This becomes particularly significant given the sensitivity of feature relevance to the specific case under consideration. The automation will optimise the identification of pertinent features, enhancing the model's effectiveness. This, in turn, enhances the depth and efficacy of decision-making processes of oracle selection under uncertainty, offering a more informed approach to smart contract development.

Expanding Security Assessment. We are going to expand our security assessment by analysing potential threats originating from various sources within the blockchain oracle ecosystem. This includes a deeper examination of adversaries beyond conventional attackers, such as competitors seeking to undermine the integrity of the oracle and external data to gain a competitive advantage or discredit the system. We will also examine the roles of third-party intermediaries involved in data transmission and processing, recognising their potential susceptibility to compromise or manipulation of data feeds to the oracle. In some cases, data providers themselves may be adversarial, intentionally providing inaccurate or manipulated data to the oracle system. By incorporating these nuanced perspectives into our security assessment approach, we aim to provide a more holistic understanding of potential threats, enabling informed decision-making and proactive mitigation strategies during the blockchain oracle platform selection process.

Further Empirical Evaluation and Development of Benchmarks. Detailed experimental results and performance benchmarks are important in evaluating the model's effectiveness. Further evaluation beyond the scope of this article will include: (i) conducting a comprehensive empirical field study catering to the wide variation of oracles, features, contexts, usage scenarios, constraints, and so on; (ii) documenting diverse performance metrics to assess various facets of the model's performance in relation to critical variables. The benchmark will also include what-if analysis support and an assessment of the diverse solutions for the said problem.

References

[1] ISO/TC 307. 2019. Blockchain and Distributed Ledger Technologies—Overview of and Interactions between Smart Contracts in blockchain and Distributed Ledger Technology Systems. Retrieved March 23, 2022 from https://www.iso.org/standard/75624.html

[2] Sabreen Ahmadjee. 2022. Oracle DSM. Retrieved March 23, 2022 from https://bitbucket.org/Smart_Contract/oracle_dsm/src/master/

[3] Sabreen Ahmadjee, Carlos Mera-Gómez, Rami Bahsoon, and Rick Kazman. 2022. A study on blockchain architecture design decisions and their security attacks and threats. *ACM Transactions on Software Engineering and Methodology* 31, 2 (Apr. 2022), Article 36e, 45 pages. DOI: https://doi.org/10.1145/3502740

[4] Sabreen Ahmadjee, Carlos Mera-Gómez, and Rami Bahsoon. 2021. Assessing smart contracts security technical debts. In *Proceedings of the IEEE/ACM International Conference on Technical Debt (TechDebt '21)*. IEEE, 6–15. DOI: https://doi.org/10.1109/TechDebt52882.2021.00010
[5] Hamda Al-Breiki, Muhammad Habib Ur Rehman, Khaled Salah, and Davor Svetinovic. 2020. Trustworthy blockchain oracles: Review, comparison, and open research challenges. *IEEE Access* 8 (2020), 85675–85685. DOI: https://doi.org/10.1109/ACCESS.2020.2992698
[6] Ayana T. Aspembitova and Michael A. Bentley. 2022. Oracles in decentralized finance: Attack costs, profits and mitigation measures. *Entropy* 25, 1 (2022), 60.
[7] Band. 2021. Band Protocol. Retrieved March 23, 2022 from https://docs.bandchain.org/whitepaper/system-overview.html
[8] BandChain. 2020. BandChain Whitepaper. Retrieved March 9, 2022 from https://docs.bandchain.org/whitepaper/system-overview.html
[9] Elena Baninemeh, Siamak Farshidi, and Slinger Jansen. 2023. A decision model for decentralized autonomous organization platform selection: Three industry case studies. *Blockchain: Research and Applications* 4 (2023), 100127.
[10] T. Bernani. 2019. Documentation of Provable. Retrieved March 2, 2022 from https://docs.provable.xyz/
[11] Emanuele Borgonovo. 2017. Sensitivity analysis: An introduction for the management scientist. In *International Series in Operations Research and Management Science*. Springer.
[12] Glenn A. Bowen. 2009. Document analysis as a qualitative research method. *Qualitative Research Journal* 9 (2009), 27–40.
[13] Giulio Caldarelli and Joshua Ellul. 2021. The blockchain oracle problem in decentralized finance—A multivocal approach. *Applied Sciences* 11, 16 (2021). 2076–3417 DOI: https://doi.org/10.3390/app11167572
[14] Yair Censor. 1977. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization* 4, 1 (1977), 41–59.
[15] Chainlink. 2020. What Is the Blockchain Oracle Problem? Retrieved March 3, 2022 from https://blog.chain.link/what-is-the-blockchain-oracle-problem
[16] Chainlink. 2021. Tellor: A decentralized Oracle. Retrieved March 8, 2022 from https://docs.tellor.io/tellor/
[17] Kuang-Hua Chang. 2015. Chapter 19 - Multiobjective optimization and advanced topics. In *e-Design*. Kuang-Hua Chang (Ed.), Academic Press, 1105–1173. DOI: https://doi.org/10.1016/B978-0-12-382038-9.00019-3
[18] Krishnendu Chatterjee, Amir Kafshdar Goharshady, and Arash Pourdamghani. 2019. Probabilistic smart contracts: Secure randomness on the blockchain. In *Proceedings of the IEEE International Conference on Blockchain and Cryptocurrency (ICBC '19)*. IEEE, 403–412.
[19] DSDM Consortium. 2014. The DSDM Agile Project Framework. Retrieved December 2, 2022 from https://www.agilebusiness.org/page/ProjectFramework_10_MoSCoWPrioritisation
[20] Juliet Corbin and Anselm Strauss. 2014. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*. Sage Publications.
[21] Adán Sánchez de Pedro, Daniele Levi, and Luis Iván Cuende. 2017. Witnet: A decentralized oracle network protocol. arXiv:1711.09756. Retrieved from https://arxiv.org/abs/1711.09756
[22] Saulo S. de Toledo, Antonio Martini, and Dag I. K. Sjøberg. 2021. Identifying architectural technical debt, principal, and interest in microservices: A multiple-case study. *Journal of Systems and Software* 177 (2021), 110968. DOI: https://doi.org/10.1016/j.jss.2021.110968
[23] Shiva Ebneyamini and Mohammad Reza Sadeghi Moghadam. 2018. Toward developing a framework for conducting case study research. *International Journal of Qualitative Methods* 17, 1 (2018), 1609406918817954.
[24] Steve Ellis, Ari Juels, and Sergey Nazarov. 2017. Chainlink: A decentralized oracle network. Retrieved March 11, 2017 from https://research.chain.link/whitepaper-v1.pdf
[25] Shayan Eskandari, Mehdi Salehi, Wanyun Catherine Gu, and Jeremy Clark. 2021. SoK: Oracles from the ground truth to market manipulation. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*. ACM, New York, NY, 127–141. DOI: https://doi.org/10.1145/3479722.3480994
[26] Siamak Farshidi. 2020. *Multi-Criteria Decision-Making in Software Production*. Ph.D. Dissertation. Utrecht University.
[27] Siamak Farshidi and Slinger Jansen. 2020. A decision support system for pattern-driven software architecture. In *Proceedings of the 14th European Conference on Software Architecture (ECSA '20)*, Tracks and Workshops. Springer, 68–81.
[28] Siamak Farshidi, Slinger Jansen, Rolf de Jong, and Sjaak Brinkkemper. 2018. A decision support system for software technology selection. *Journal of Decision Systems* 27, sup1 (2018), 98–110. DOI: https://doi.org/10.1080/12460125.2018.1464821
[29] Siamak Farshidi, Slinger Jansen, Sergio España, and Jacco Verkleij. 2020. Decision support for blockchain platform selection: three industry case studies. *IEEE Transactions on Engineering Management* 67, 4 (2020), 1109–1128. DOI: https://doi.org/10.1109/TEM.2019.2956897

[30] Siamak Farshidi, Slinger Jansen, and Sven Fortuin. 2021. Model-driven development platform selection: Four industry case studies. In *Software and Systems Modeling*, 1525–1551.
[31] Siamak Farshidi, Izaak Beer Kwantes, and Slinger Jansen. 2023. Business process modeling language selection for research modelers. In *Software and Systems Modeling*, 137–162.
[32] Valentina Gatteschi, Fabrizio Lamberti, Claudio Demartini, Chiara Pranteda, and Víctor Santamaría. 2018. Blockchain and smart contracts for insurance: Is the technology mature enough? *Future Internet* 10, 2 (2018), 20.
[33] Ammar Hassan, Imran Makhdoom, Waseem Iqbal, Awais Ahmad, and Asad Raza. 2023. From trust to truth: Advancements in mitigating the Blockchain Oracle problem. *Journal of Network and Computer Applications* 217 (2023), 103672.
[34] Jonathan Heiss, Jacob Eberhardt, and Stefan Tai. 2019. From oracles to trustworthy data on-chaining systems. In *Proceedings of the IEEE International Conference on Blockchain (Blockchain '19)*. IEEE, 496–503. DOI: https://doi.org/10.1109/Blockchain.2019.00075
[35] iExec. 2022. iExec Technical Documentation. Retrieved March 9, 2022 from https://docs.iex.ec/
[36] Clemente Izurieta and Mary Prouty. 2019. Leveraging secdevops to tackle the technical debt associated with cybersecurity attack tactics. In *Proceedings of the IEEE/ACM International Conference on Technical Debt (TechDebt '19)*. IEEE, 33–37.
[37] Keywordseverywhere. 2022. Keywords Every Where. Retrieved March 27, 2022 from https://keywordseverywhere.com/
[38] Javed Ali Khan, Izaz Ur Rehman, Yawar Hayat Khan, Iftikhar Javed Khan, and Salman Rashid. 2015. Comparison of requirement prioritization techniques to find best prioritization technique. *International Journal of Modern Education & Computer Science* 7, 11 (2015), 53–59.
[39] Bowen Liu, Pawel Szalachowski, and Jianying Zhou. 2021. A first look into DeFi oracles. In *Proceedings of the IEEE International Conference on Decentralized Applications and Infrastructures (DAPPS '21)*, 39–48. DOI: https://doi.org/10.1109/DAPPS52256.2021.00010
[40] Sin Kuang Lo, Xiwei Xu, Mark Staples, and Lina Yao. 2020. Reliability analysis for blockchain oracles. *Computers and Electrical Engineering* 83, 10658 (2020), 2.
[41] Weisheng Lu, Xiao Li, Fan Xue, Rui Zhao, Liupengfei Wu, and Anthony G. O. Yeh. 2021. Exploring smart construction objects as blockchain oracles in construction supply chain management. *Automation in Construction* 129 (2021), 103816.
[42] Loi Luu, Duc-Hiep Chu, Hrishi Olickel, Prateek Saxena, and Aquinas Hobor. 2016. Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 254–269.
[43] Mrinmoy Majumder. 2015. Multi criteria decision making. In *Impact of Urbanization on Water Shortage in Face of Climatic Aberrations*. Springer, 35–47.
[44] Kamran Mammadzada, Mubashar Iqbal, Fredrik Milani, Luciano García-Bañuelos, and Raimundas Matulevičius. 2020. Blockchain oracles: A framework for blockchain-based applications. In *Proceedings of the International Conference on Business Process Management*. Springer International Publishing, 19–34.
[45] Antonio Martini, Jan Bosch, and Michel Chaudron. 2014. Architecture technical debt: Understanding causes and a qualitative model. In *Proceedings of the 2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, 85–92. DOI: https://doi.org/10.1109/SEAA.2014.65
[46] Ilhaam A. Omar, Haya R. Hasan, Raja Jayaraman, Khaled Salah, and Mohammed Omar. 2021. Implementing decentralized auctions using blockchain smart contracts. *Technological Forecasting and Social Change* 168 (2021), 120786. DOI: https://doi.org/10.1016/j.techfore.2021.120786
[47] Pim Otte, Martijn de Vos, and Johan Pouwelse. 2017. TrustChain: A Sybil-resistant scalable blockchain. *Future Generation Computer Systems* 107 (2017), 770–780.
[48] Owasp. 2019. OWASP Risk Rating Methodology. Retrieved March 23, 2022 from https://owasp.org/www-community/OWASP_Risk_Rating_Methodology
[49] Paralink. 2021. Paralink Network. Retrieved December 20, 2022 from https://cutt.ly/0DPtJY4
[50] Amirmohammad Pasdar, Zhongli Dong, and Young Choon Lee. 2021. Blockchain oracle design patterns. arXiv:2106.09349. Retrieved from https://arxiv.org/abs/2106.09349
[51] Amirmohammad Pasdar, Young Choon Lee, and Zhongli Dong. 2023. Connect API with blockchain: A survey on blockchain oracle implementation. *ACM Computing Surveys* 55, 10 (Feb. 2023), Article 208, 39 pages. DOI: https://doi.org/10.1145/3567582
[52] PeckShield. 2020. Cheese Bank Incident: Root Cause Analysis. Retrieved December 20, 2022 from shorturl.at/zDX38
[53] Simone Porru, Andrea Pinna, Michele Marchesi, and Roberto Tonelli. 2017. Blockchain-oriented software engineering: Challenges and new directions. In *Proceedings of the IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C '17)*, 169–171. DOI: https://doi.org/10.1109/ICSE-C.2017.142
[54] Niall Roche and Alastair P. Moore. 2020. *Oraclised Data Schemas: Improving Contractual Certainty in Uncertain Times*. Ph.D. Dissertation. London University; UCL Centre for Blockchain Technologies.

[55] Per Runeson and Martin Höst. 2009. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering* 14, 2 (2009), 131–164.
[56] Stephanie. 2014. What Is Cohen's Kappa Statistic? Retrieved May 16, 2022 from https://www.statisticshowto.com/cohens-kappa-statistic/
[57] R. Stevens. 2020. After DeFi Lost 100 Million to Flash Loan Attacks, Curve Pushes Chainlink. Retrieved April, 2024 from https://decrypt.co/49758/after-100-million-lost-to-flash-loan-attacks-curve-pushes-chainlink
[58] Julien Thevenard. 2019. Decentralised Oracles: A Comprehensive Overview. Retrieved December 10, 2022 from shorturl.at/mxFL7
[59] Evangelos Triantaphyllou, Bo Shu, S. Nieto Sanchez, and Tony Ray. 1998. Multi-criteria decision making: An operations research approach. In *Encyclopedia of Electrical and Electronics Engineering*, 175–186.
[60] Hanny Tufail, Iqra Qasim, Muhammad Faisal Masood, Sara Tanvir, and Wasi Haider Butt. 2019. Towards the selection of optimum requirements prioritization technique: A comparative analysis. In *Proceedings of the 5th International Conference on Information Management (ICIM '19)*, 227–231. DOI: https://doi.org/10.1109/INFOMAN.2019.8714709
[61] Z. Wahid and N. Nadir. 2013. Improvement of one factor at a time through design of experiments. *World Applied Sciences Journal* 21, 1 (2013), 56–61.
[62] Niall Roche Walter Hernandez. 2020. Oracle Data Lexicon—Bringing Contractual Certainty in Uncertain Time. Retrieved March 23, 2022 from https://challenge.globallegalhackathon.com/gallery/5ec84aef202da60044c03d6b
[63] Niall Roche Walter Hernandez. 2020. An Oracle to Allow Pandemic-Aware Policies. Retrieved March 23, 2022 from https://devpost.com/software/covidhack-oracle-provable
[64] Bin Wang, Xiaohan Yuan, Li Duan, Hongliang Ma, Chunhua Su, and Wei Wang. 2022. DeFiScanner: Spotting DeFi attacks exploiting logic vulnerabilities on blockchain. *IEEE Transactions on Computational Social Systems* 11, 2 (2022), 1577–1588.
[65] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer Science & Business Media.
[66] Gavin Wood. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper* 151, 2014 (2014), 1–32.
[67] Xiwei Xu, H. M. N. Dilum Bandara, Qinghua Lu, Ingo Weber, Len Bass, and Liming Zhu. 2021. A decision model for choosing patterns in blockchain-based applications. In *Proceedings of the IEEE 18th International Conference on Software Architecture (ICSA '21)*, 47–57. DOI: https://doi.org/10.1109/ICSA51549.2021.00013
[68] Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. 2019. Blockchain technology overview. arXiv:1906.11078. Retrieved from https://arxiv.org/abs/1906.11078
[69] Xin-She Yang. 2014. Chapter 14—Multi-objective optimization. In *Nature-Inspired Optimization Algorithms*. Xin-She Yang (Ed.), Elsevier, 197–211. DOI: https://doi.org/10.1016/B978-0-12-416743-8.00014-2
[70] Fan Zhang, Ethan Cecchetti, Kyle Croman, Ari Juels, and Elaine Shi. 2016. Town crier: An authenticated data feed for smart contracts. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. ACM, New York, NY, 270–282. DOI: https://doi.org/10.1145/2976749.2978326
[71] Yahui Zhang, Min Zhao, Tingquan Li, and Huan Han. 2020. Survey of attacks and defenses against SGX. In *Proceedings of the IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC '20)*. IEEE, 1492–1496. DOI: https://doi.org/10.1109/ITOEC49072.2020.9141835
[72] Weiqin Zou, David Lo, Pavneet Singh Kochhar, Xuan-Bach Dinh Le, Xin Xia, Yang Feng, Zhenyu Chen, and Baowen Xu. 2021. Smart contract development: Challenges and opportunities. *IEEE Transactions on Software Engineering* 47, 10 (2021), 2084–2106. DOI: https://doi.org/10.1109/TSE.2019.2942301

Appendix A

A complete mapping between *a.* the Boolean oracle PF and *b.* the Boolean oracle features and quality attributes.

Table A1. Mapping between the Boolean Oracle PF

Boolean Oracle Criteria and Platform	P1	P2	P3	P4	P5	P6	P7	P8
Type of Data Feeder								
Centralised	1	1	0	0	0	0	0	0
Semi-decentralised	0	0	1	1	0	1	1	0
Fully decentralised	0	0	0	0	1	0	0	1
Type of Data Source								
Single	1	1	1	1	1	1	1	1
Multiple	0	1	1	1	1	1	1	1
Data Validation Mechanism								
Trusted third party	1	0	0	0	0	0	0	0
Trusted Execution Environment	0	1	1	0	0	0	0	1
Consensus	0	0	1	1	1	1	1	1
Integration Methods								
On-chain	0	0	1	1	1	1	1	0
Off-chain	1	1	0	0	0	0	0	0
Hybrid	0	0	0	0	0	0	0	1
Encryption Method								
Symmetric encryption	0	0	0	0	0	0	0	0
Asymmetric encryption	1	1	1	0	0	0	0	1
Data Feeders Selection Method								
Staking	0	0	1	0	1	1	1	1
Reputation	0	0	1	1	0	1	1	1
PoW	0	0	0	1	1	0	0	0
PoCo	0	0	0	0	0	0	0	1
Random	0	0	0	0	1	0	1	1
Pseudo-random	0	0	0	1	0	0	0	0
Aggregation Mechanism								
Statistical Measure	0	0	1	0	0	0	1	0
Mean	0	0	0	1	0	1	0	0
Median	0	0	0	1	1	1	0	0
Mode	0	0	0	1	0	0	0	0
Voting	0	0	0	0	0	0	0	1
Dispute Resolution								
Voting	0	0	0	0	1	0	0	0
Staking	0	0	0	0	1	0	1	0
Statistical Measure	0	0	1	0	0	0	1	0
Incorrect Data								
Corrected	0	0	0	0	1	0	0	0
Reverted	0	0	0	0	0	0	0	0
Incentive Scheme								
Reward	0	0	1	1	1	1	1	1
No reward	1	1	0	0	0	0	0	0
Punishment Methods								
Slash	0	0	1	1	1	1	1	1
Ban	0	0	0	0	1	0	0	0
Reputation Loss	0	0	1	1	0	1	1	1
Native Token (NT)								
Exist	0	0	1	1	1	1	1	1
Not exist	1	1	0	0	0	0	0	0
Using NT for Requesting								
Required	0	0	1	0	1	1	1	1
Not-required	1	1	0	1	0	0	0	0

Table A2. Mapping between the Boolean Oracle PF

Boolean Oracle Criteria and Attribute	C	I	A	NR	Auth	Trust	T	LL	LC	LTFC	Acc
Type of Data Feeder											
Centralised	0	0	0	0	0	0	0	1	0	0	0
Semi-decentralised	0	0	1	0	0	0	0	0	0	0	0
Fully decentralised	0	1	1	0	0	1	0	0	0	0	0
Type of Data Source											
Single	0	0	0	0	0	0	0	1	0	0	0
Multiple	0	1	1	0	0	1	0	0	0	0	0
Data Validation Mechanism											
Trusted third party	1	1	0	1	1	0	0	1	0	0	0
Trusted Execution Environment	1	1	0	1	1	0	0	1	0	0	0
Consensus	0	1	0	1	1	1	0	0	0	0	0
Integration Methods											
On-chain	0	1	1	0	0	0	1	0	0	0	0
Off-chain	0	0	0	0	0	0	0	0	1	0	0
Hybrid	0	1	0	0	0	1	1	0	1	0	0
Encryption Method											
Symmetric encryption	1	0	0	0	0	0	0	0	0	0	0
Asymmetric encryption	1	0	0	0	0	0	0	0	0	0	0
Data Feeders Selection Method											
Staking	0	1	0	0	0	0	0	0	0	0	0
Reputation	0	1	0	0	0	0	0	0	0	0	0
PoW	0	1	0	0	0	0	0	0	0	0	0
PoCo	0	1	0	0	0	0	0	0	0	0	0
Random	0	1	0	1	1	0	0	0	0	0	0
Pseudo-random	0	1	0	1	1	0	0	0	0	0	0
Aggregation Mechanism											
Statistical Measure	0	0	0	0	0	0	0	1	1	0	0
Mean	0	0	0	0	0	0	0	1	1	0	0
Median	0	0	0	0	0	0	0	1	1	0	0
Mode	0	0	0	0	0	0	0	1	1	0	0
Voting	0	0	0	0	0	0	0	0	0	0	0
Dispute Resolution											
Voting	0	0	0	0	0	0	0	0	0	0	0
Staking	0	1	0	0	0	0	0	0	0	0	0
Statistical Measure	0	0	0	0	0	0	0	1	1	0	0
Incorrect Data											
Corrected	0	0	0	0	0	0	0	0	0	0	0
Reverted	0	0	0	0	0	0	0	1	0	0	0
Incentive Scheme											
Reward	0	1	0	0	0	0	0	0	0	0	0
No reward	0	0	0	0	0	0	0	0	0	1	0
Punishment Methods											
Slash	0	1	0	0	0	0	0	1	0	0	0
Ban	0	1	0	0	0	0	0	1	0	0	0
Reputation Loss	0	1	0	0	0	0	0	1	0	0	0
Native Token (NT)											
Exist	0	0	0	0	0	0	0	0	0	0	0
Not exist	0	0	0	0	0	0	0	0	0	0	0
Using NT for Requesting											
Required	0	0	0	0	0	0	0	0	0	0	0
Not-required	0	0	0	0	0	0	0	0	0	0	1

Received 22 November 2023; revised 20 July 2024; accepted 6 September 2024