1-1-2023

Using Blockchain for Enabling Transparent, Traceable, and Trusted University Ranking Systems

Ammar Battah
Khalifa University of Science and Technology

Khaled Salah
Khalifa University of Science and Technology

Raja Jayaraman
Khalifa University of Science and Technology

Ibrar Yaqoob
Khalifa University of Science and Technology

Ashraf Khalil
Zayed University, ashraf.khalil@zu.ac.ae

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

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

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Using Blockchain for Enabling Transparent, Traceable, and Trusted University Ranking Systems

AMMAR BATTAH[1], KHALED SALAH[2], RAJA JAYARAMAN[1], IBRAR YAQOOB[2], ASHRAF KHALIL[3]

[1]Department of Industrial and Systems Engineering, Khalifa University, Abu Dhabi 127788, UAE.
[2]Department of Electrical Engineering and Computer Science, Khalifa University, Abu Dhabi 127788, UAE.
[3]College of Technological Innovation, Zayed University, Abu Dhabi 144534, UAE.

ABSTRACT Ranking systems have proven to improve the quality of education and help build the reputation of academic institutions. Each of the current academic ranking systems is based on different methodologies, criteria, and standards of measurement. Academic and employer reputations are subjective indicators of some rankings determined through surveying that is neither transparent nor traceable. The current academic ranking systems fall short of providing transparency and traceability features for both subjective and objective indicators that are used to calculate the ranking. Also, the ranking systems are managed and controlled in a centralized manner by specific entities. This raises concerns about fairness and trust. In this paper, we propose a blockchain-based solution to enable transparent, traceable, trusted, and decentralized academic ranking systems. We develop smart contracts to automatically govern entity interactions according to set policies. We leverage decentralized storage, oracles, and threshold encryption to securely fetch, store, and share institutions' ranking data. We present algorithms along with their implementation and testing details, as well as validate smart contracts. The proposed solution is evaluated in terms of cost, throughput and latency, and security to show its affordability, resiliency against attacks, and superiority. All developed smart contract codes are made publicly available on GitHub.

INDEX TERMS University Ranking, Blockchain, Ethereum, Smart Contract, Trust, Security, Traceability

I. INTRODUCTION

Rankings help academic institutions improve the quality of education and build their reputations. The most well-known ranking systems are Quacquarelli Symonds (QS), Times Higher Education (THE), and Academic Ranking of World Universities (ARWU). Each of the ranking systems has different criteria, leading to discrepancies in the ranking of Higher Education Institutions (HEIs) [1], [2]. The subjectivity of the rankings raises questions. For instance, 50% of the overall QS score relies on surveys, which constitute 33% of THE's overall score as well [3]. Compared to rankings that rely on subjective indicators, the ARWU uses more objective metrics, such as achieving a Nobel Prize or publishing in specific journals such as Nature and Science [4]. However, even an objective-based ranking such as ARWU is criticized on grounds of lack of transparency, limited assessment in comparison to other rankings, and bias towards older and larger institutions. In general, all the current academic ranking systems are centralized and do not provide full transparency and traceability, which impacts the trustworthiness of the results [5].

Transparency is crucial to build trust among users towards rankings. Details such as whom the survey pool consists of, the response rate, where they are based, and if there is any potential bias help to build up such trust [6]. However, surveys utilized by ranking systems are not published nor is the publicly information shared [7]. This leads to bias such as that observed in [8], where universities that utilized QS consultancy services climbed the rankings with no apparent institutional changes that justified the change. Because most ranking systems are centralized, it is common for people to use their power for personal gain [9], [10]. Furthermore, in several cases, the aggregated scores differed from what they were claimed to be, yielding misleading results [11], [12]. The centralization of ranking bodies also leads to potential bias based on the location of the ranking body. Rankings that

rely on subjective sources must be transparent with the data and process of generating the indicator, but current systems fall short on this requirement.

Because of the need for more accurate ranking systems, national rankings for each country and new global rankings like Webometrics and Leiden have emerged. Each of the rankings implements different methodologies [13]–[16]. In addition, some analysts have constructed alternative rankings, such as U-Multirank, which offer a multi-dimensional view of HEIs. U-Multirank offers transparency by showing the individual metrics that could make up an HEI's overall rank or score. While there have been many alternative rankings, there needs to be a solution to tackle the issues related to centralization, trust, traceability, and transparency. To the best of our knowledge, a decentralized platform has yet to be proposed to remove the single point of trust given to the entity; participants instead place trust in the functionality of the network. In such a system, concerns about bias, manipulation, and transparency can be addressed with confidence.

In this paper, we propose a blockchain-based solution that aims to enhance university ranking systems by offering transparency, traceability, trust, and audit features. The blockchain is a hash chain composed of blocks containing many transactions. Each block contains the hash of the previous block; any change to it invalidates the following blocks since their hashes are also altered. This allows network participants to transparently monitor the transactions on the network without concern for manipulation. Furthermore, since ranking systems rely on different indicators from different sources, the blockchain requires means to interact with the outside world. In our proposed approach, a decentralized oracle subsystem is used to handle external interactions (off-chain). Smart contracts are developed to manage the oracles and ensure their results are valid. Moreover, smart contracts enable decentralized governance of interactions in a transparent, secure, and autonomous manner. Users can validate the authenticity of calculations and external results that are recorded through oracle consensus. Through such an approach, several sources can be relied on to aid in computing the chosen indicators while maintaining transparency from the data source to the representation. Furthermore, the anonymous participants can be authenticated and held accountable to prevent malicious activities in the ranking process.

The main contributions of the paper are as follows:

- We propose a blockchain-based solution to enable transparent, traceable, and trusted university ranking systems. We present the proposed system architecture, the sequence of interactions, and the role of each entity. We also provide guidelines for interpreting the data produced by the ranking systems.
- We design an approach governed by smart contracts that enforces the required policies and transparently records the interactions. We explain in detail how to register stakeholders and collect ranking data with accountability.
- We utilize Oracle clusters to conduct anonymous surveys, maintaining decentralization and transparency throughout the approach.
- We secure and scale the proposed solution through threshold encryption of Oracle transactions, maintaining traceability and transparency.
- We employ and provision a decentralized storage space under the governance of oracles and smart contracts to trace data from HEI's, surveys and online sources.
- We implement a proof-of-concept smart contract, as well as present the algorithms along with their testing and validation details. The smart contract codes are publicly made available on Github[1].
- We evaluate the developed smart contracts through a gas cost analysis, gauging performance by the throughput of transactions, and analyzing their security parameters.

The remainder of the paper is organized as follows. Section II presents the related work. Section III discusses the proposed approach and its components, as well as highlights the system architecture and sequence of interactions. Section IV presents the implementation, testing, and validation details of the proposed solution. Section V provides cost, security, and performance analyses of the proposed solution. We provide concluding remakes along with future work in Section VI.

II. RELATED WORKS

In this section, we review the existing literature focused on university ranking systems, as well as general advancements made in the field of education.

A. EDUCATIONAL SECTOR ADVANCEMENTS

The importance of the education sector cannot be neglected, and various efforts have been put toward advancing different educational areas [17], [18], including ranking systems. However, while traditional university ranking systems have received much criticism, they are still in bloom with limited alternatives being offered. The International Ranking Expert Group (IREG) has compiled a list of the various university rankings under the Inventory of International Rankings [19]. Despite the criticisms, there is still limited research for alternative approaches to the traditional centralized rankings. The current ranking systems have been critiqued, and guidelines have been suggested. The researchers in [20] proposed a framework to evaluate global rankings independently, rather than devising a ranking system. The method follows the devised SCOPE framework, which takes into consideration: 1) what is valued about the entity; 2) the importance of the evaluative context; 3) evaluation options; 4) a deep examination of the evaluation approach; and 5) the requirement that an external entity evaluate the ranking. The work also evaluates the common ranking systems based on a criterion that follows the proposed framework. While the proposed framework is community-driven, it is too broad and needs

[1]https://github.com/anonymousgitter20221206/Decentralized-Ranking

to cover more bases for the average user to assess ranking systems appropriately.

The blockchain is slowly being incorporated into solutions to tackle issues posed by current approaches. The authors in [21] leveraged the blockchain to keep track of professional skills, education, and employment history. Such information requires verification, which network users supply in connection with incentives based on the Vickrey-Clarke-Groves (VCG) incentive mechanism. Authors in [22] proposed a decentralized ranking system for educational content. The solution relies on verified subject matter experts (SMEs) to rate the online content. While advancements are occurring, the current university ranking is still unaffected.

In our approach, we utilize a public blockchain to provide equal access to all entities. On the other end of the spectrum, centralized solutions violate the main requirements of trust, transparency, traceability, and auditability, as mainly the controlling entity has access to and control over the central server or database. Aside from the traditional centralized databases, there have been recent proposals for different centralized ledgers to confront the blockchain. LedgerDB is a prominent example, which proposes a centralized ledger that can be audited and verified through a timestamp authority (TSA) [23]. While Aquareum utilizes the blockchain simultaneously with a centralized ledger to enhance its capabilities [24]. An intermediate solution would be private blockchains, such as Hyperledger Fabric, which would be suitable for consortiums and enterprises [25]. While the mentioned solutions commonly offer higher throughput, they do not scale with participants. Furthermore, the mentioned solutions assume trust in the controlling entity, whether a third party or a consortium, and restrict the participation and access of other entities.

B. CURRENT RANKING SYSTEMS

The most renowned ranking systems are QS, THE, and ARWU. Some other global rankings include Webometrics, Leiden University Ranking, and U-Multirank. The aforementioned rankings systems are either centralized or governed by a consortium. Due to the lack of decentralized approaches for university ranking, we will focus on the current utilized approaches.

1) THE

THE ranking was established in 2004 and has focused on research-intensive institutions. The bibliometric data needed for ranking is extracted from Elsevier's Scopus database. Furthermore, the indicators it uses encompass five areas, each weighted differently: teaching (30%), research (30%), citations (30%), international outlook (7.5%), and industry income (2.5%). The weight distribution of the ranking system heavily focuses on research, amounting to 60% if we consider citations as a compartment of research. Moreover, the accuracy of the chosen indicators in representing the area needs to be verified. For instance, 18% of research is based on a reputation survey, which is a subjective indicator. Research income and productivity contribute to 6% each, which makes it difficult to prove a direct correlation with quality research. This also applies to citations, which comprise a large portion of the overall score. While the first 200 institutions have an individual rank, institutions above 200 are given a banded ranking (i.e., 201-205).

2) QS

The other prominent global ranking is by QS, which was also founded in 2004. THE and QS published joint rankings until 2009. As for the bibliometric data, QS also relies on Elsevier's Scopus database. Eight indicators of various weight contribute to the QS ranking: academic reputation, employer reputation, faculty/student ratio, citations per faculty, international student ratio, international faculty ratio, international research network, and employment outcomes. The indicators regarding academic reputation and employer reputation are based on surveys conducted by QS, which account for 50% of the overall score. Surveys are conducted based on a collection of 130,000 expert opinions within the higher education space. Again, questions are raised due to a lack of full transparency, especially when a large portion of the overall score consists of subjective indicators. QS collates experts through submitted contact lists from institutions, sign-ups, and the IBIS database. Regarding rankings, institutions in the top 500 are given individual ranks, while those above are in bands of 10, 50, and 200, increasing the lower the rank.

3) ARWU

Shanghai Jiao Tong University created the first global ranking, Academic Ranking of World Universities (ARWU), in 2003. Currently, it is managed by Shanghai Consultancy, which focuses on six objective indicators in four areas. The four areas of focus are quality of education, quality of faculty, research output, and per capita performance. These areas account for 10%, 40%, 40% and 10%, respectively, of the total weight. ARWU is unique in not relying on subjective indicators and focusing specifically on objective indicators. At the same time, it is criticized for not including essential aspects of evaluating universities such as teaching, social work, employability, and internationalization [6], [26]. Furthermore, weight selection is highly subjective, as 30% of the weight is given to alumni or staff of an institution that has won a Nobel prize or a field medal, both of which are rare. ARWU collects data through official websites such as the Nobel Prize website and relies on the Clarivate database.

4) Webometrics

Webometrics rankings began in 2004 at Cybermetrics Lab and currently rank 12,000 institutions. Compared to the above-mentioned rankings, Webometrics is more comprehensive in terms of included institutions. Its ranking methodology differs in that it includes both webometric and bibliometric indicators. The reasoning for the different approach is to encourage academic web presence to facilitate the exchange of scientific and cultural knowledge. The three main

indicators are web content impact, top-cited researchers, and top-cited papers, with a weight of 50%, 10% and 40%, respectively. Furthermore, the ranking is carried out twice a year using public data, making it more transparent than other rankings. However, this creates a flaw where institutions could focus on webometrics, increasing their web presence, to overcome other institutions with a weaker presence.

5) Leiden

The Leiden University Ranking, published by Leiden University, is the only ranking managed by a university. It is emphasized that the ranking is based on the Web of Science database, with Leiden handling the data enrichment. The ranking focuses only on research and extracting and enriching bibliometric data. The main indicators are scientific impact, collaboration, open access, and gender. The ranking introduces features such as considering the proportion of publications in the top percentiles of the field and fractional counting (i.e., a fraction for co-authoring, fraction for a non-core publication). An advantage of the ranking is the representation of rankings, where rankings are listed according to individual indicators rather than with a single composite score. Furthermore, there is no subjectivity since only bibliometric data is used, which is considered more transparent than other rankings with its included documentation. However, a disadvantage is that Leiden treats educational institutes as research centers, only considering their research aspects.

6) U-Multirank

U-Multirank takes a different approach to HEI evaluations that aims to "rectify" the current ranking systems. It is proposed by a European Union consortium that includes the German Center for Higher Education (CHE), the Netherlands Center for Higher Education Policy Studies (CHEPS), Leiden's Centre of Science and Technology Studies, and several reputable stakeholders. U-Multirank has no composite value to rank HEIs but rates them based on categories. Five dimensions are considered, and five categories from A to E are used for categorizing HEIs [27]. The ranking focuses on being multi-dimensional and user-driven, which other rankings lack. The five dimensions under which the indicators fall include (1) teaching and learning, (2) research, (3) knowledge transfer, (4) international orientation, and (5) regional engagement. The unique nature of U-Multirank creates no single winner; rather than differentiating between the institutions, several winners are categorized as "A" (very good). While this makes it less competitive for institutions and decreases the drive to be on top, it gives institutions space to excel in their own way in different dimensions, as opposed to the rigid ranking approaches that set what an exemplary HEI should be. Users can find suitable institutions based on their needs rather than the needs dictated by the one-dimensional ranking. This user-driven approach allows institutions to excel at what they do and achieve the appropriate recognition and interest.

While the U-Multirank approach is getting recognition for being a step in the right direction, it is still a work in progress. It is proposed by a consortium of entities, which might introduce bias or a limited view of the educational landscape. The majority of participants are from European institutions, which are present at least four times more than other regions [28]. The consortium has to maintain the service by governing and funding it, which might prove difficult and introduce doubts as external entities are involved. For instance, Web of Science (WoS) was chosen for bibliometric data, even though it could be argued that using Scopus or both of them is more suitable. As such, decisions are taken based on the consortium's views. This concern is magnified by the lack of full transparency, as U-Multirank checks data for consistency, missing data, and plausibility [28].

III. PROPOSED BLOCKCHAIN-BASED SOLUTION FOR UNIVERSITY RANKING SYSTEMS

In this section, we present our proposed blockchain-based solution that enhances the current university ranking systems. We discuss our system design and its architecture, as well as present the interaction details between the ranking system entities. Figure 1 shows an overview of system entities, highlighting the interactions for authenticating HEIs, curating data, conducting surveys, and validating their results. We propose the solution based on a public permission-less blockchain, as private permissioned blockchains limit access which is required for such a proposed system.

A. SYSTEM DESIGN AND ARCHITECTURE

A blockchain-based solution is proposed to ensure that the ranking process is decentralized, transparent, traceable, auditable, and trustworthy. On top of the integrated blockchain network, smart contracts are used to govern entity interactions according to set policies. The smart contracts provide the means to enforce the authenticity of transactions, while the blockchain ensures their integrity, transparency, and auditability. Oracles are employed to register and have their data validated by the smart contracts after they execute their tasks. The oracles are used to conduct surveys and fetch data from open sources. To address the issue of limited data storage, we use a decentralized storage system. The storage system maintains the accumulated data gathered for HEIs.

Entities in the proposed solution register only when necessary. All participants have to be part of the network with a valid address to carry out transactions on the blockchain. Each entity and its task are as follows:

- **Higher Education Institutions (HEI)**: An HEI must register before it can be rated, and an authentication process takes place for each registration request. Furthermore, HEIs should provide a path containing information relevant to the institution, such as the number of faculty, students, research centers, staff, events held, awards gained, facilities, equipment, and any relevant information that illustrates the institution's goals and capabilities. Moreover, part of the information is the

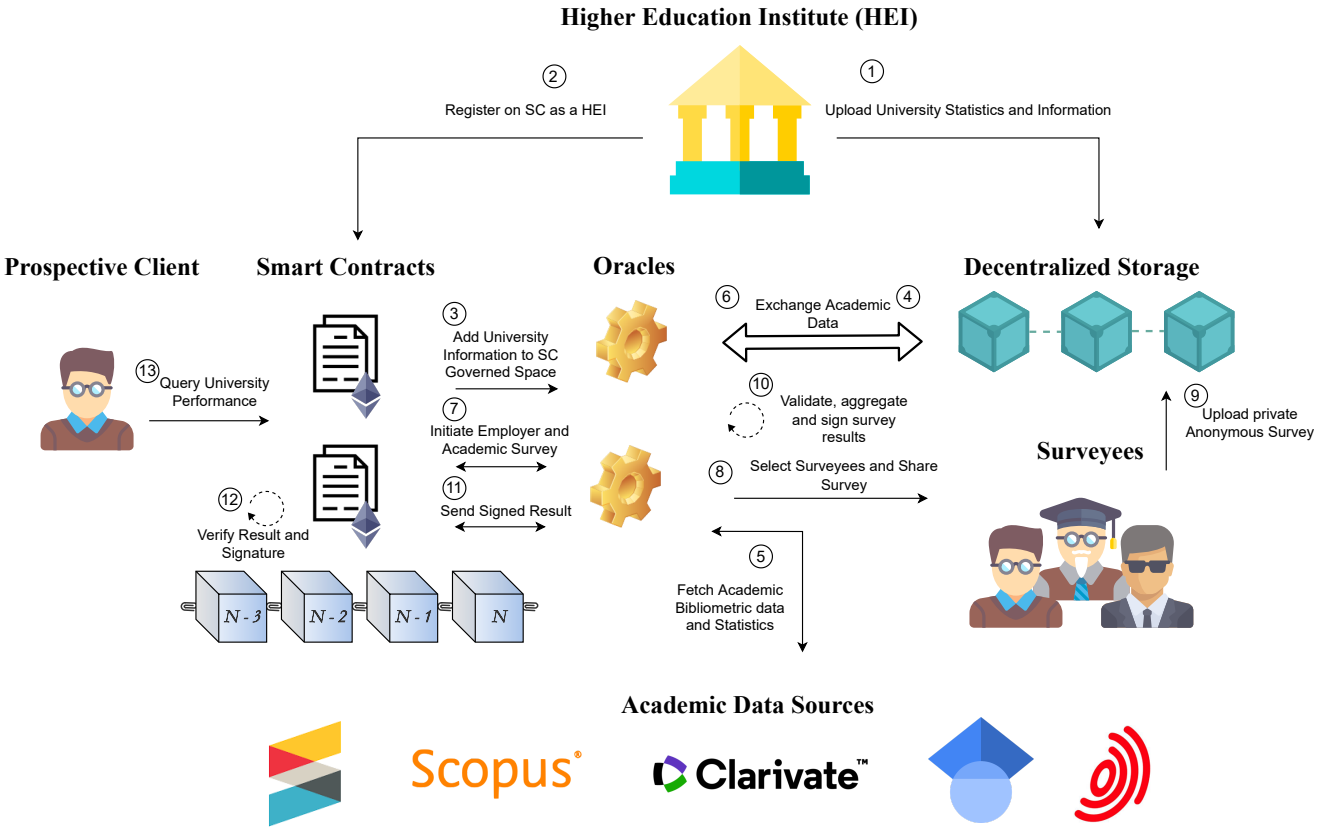

FIGURE 1. Overview of system entities, highlighting the interactions for authenticating HEIs, curating data, conducting surveys, and validating their results.

anonymous faculty's public keys, which can be used for validation by the network. Unlike other entities to be mentioned, HEIs are not anonymous and can be publicly recognized. The smart contract authenticates an HEI by validating that the Ethereum address corresponds to the registering institute. To achieve this, the registering institute needs to provide the account address on its website under its valid SSL certificate, which the oracles will check. Taking such an approach leverages the existing certificate system, ensuring that the HEI is the one providing the EA.

- **Clients**: A client is any entity interested in gauging an HEI's performance based on selected aspects. Such entities may include prospective students, employers, recruiters, researchers/lecturers, media, regulatory authorities, and national authorities. Clients are not required to register, considering their varying availability and computing capabilities. Furthermore, their communication gateway with the network and its services would be the decentralized application (DApp). The interpretation of the indicators is dependent on the client's needs. For instance, a chemical engineering student would be interested in how well the chemistry departments of each HEI rank in terms of equipment and facilities, research productivity, faculty qualifications, teaching quality, and any additional relevant indicators.
- **Smart Contracts**: Smart contracts are the mediators and governors of the network that replace the common third party in traditional solutions. They can be seen as a computerized transaction protocol that executes the terms of a contract [29]. The contract are responsible for maintaining the devised access control policies in the proposed architecture. Moreover, the smart contracts manage oracles and validate their transactions.
- **Surveyees**: Surveyees are end-users that ideally have a low load in terms of communication, computation, and storage. They do not need to register or have an EA. Surveyees who contribute to the academic survey include reputable academics such as professors, researchers, and scholars. Likewise, employers and graduate recruiters are surveyed for the employer survey. Surveyees are contacted by smart contract-governed oracles for the survey. Since survey results might be influenced by the surveyee's identity being revealed, survey submissions are anonymous.
- **Oracles**: While smart contracts govern on-chain interactions, off-chain activities are carried out by oracles. Oracles are responsible for validating HEI registration, sending, receiving, and validating surveys, and aiding in the amalgamation of academic information in the decen-

tralized storage. The decentralized oracles are grouped into clusters that communicate using the peer-to-peer (P2P) protocol to achieve their tasks in a trustworthy manner through consensus. Furthermore, the oracles act as a second layer to the blockchain by using threshold signatures. Finally, a different type of oracle is used to generate a nonce for use between oracles and surveyees.

- **Storage and Data Sources**: Off-chain storage is needed to tolerate all the ongoing transactions of the system without overloading the participants. We use the Interplanetary File System (IPFS) as a storage medium to conduct surveys between oracles and surveyees. Furthermore, the IPFS hosts the databases of HEIs, Oracles, and academics. These different entities collaborate to accumulate data and validate it when necessary. Accessible data is fetched by oracles when needed from sources such as Scopus, Clarivate, Google Scholar, PATSAT, CrossRef, Orcid, Researchgate, and similar relevant sources that provide academic data. Such data can be used to cross-check data provided by other entities to ensure the constructed database's validity.

B. ORACLES AND ENCRYPTION

The rapid advances in blockchain technology have led it to offer more features. Scalability is a major concern when implementing blockchain solutions. Our proposed approach relies on oracles to handle external interactions that cannot be carried out by the blockchain. However, external tasks are not recorded on the blockchain.

To maintain trust throughout the entire process, oracles commonly submit their results to the smart contract to achieve consensus. As such, decentralization is maintained even at the oracle level, and consensus is held on the blockchain, which can be audited. However, this becomes infeasible with the high number of oracles that are consistently used. While users require trustworthy transactions, it is not required for each oracle to see every detail of every submission. An implementation that complements such a concept is state channels [30]. Instead of multiple transactions, only two transactions are required to open and close a channel while the two concerned entities interact off the chain. To maintain trust on the chain, the entities use multi-signatures to get a consensus from the participating entities on the validity of the transaction. The multi-signature contract validates the signature before accepting it.

We employ a similar concept; however, unlike the common two-party settlement, we leverage utilization for oracle consensus. Furthermore, we use a threshold signature scheme that enables a single transaction to attain consensus after it has been initiated. On the contrary, typical multi-signatures require each individual entity to sign the transaction, creating the need for multiple verifications. This extends to key refresh, where only a shared public key is sent by the participating group. Threshold encryption is a subset of multiparty computation that allows operation execution even in an untrusted setting without revealing private data. Such an approach is crucial as it reduces the number of transactions on the chain, thereby reducing cost and increasing scalability. The threshold determines how many m oracles are required to generate a valid signature out of the total cluster n. In the proposed solution, oracles are grouped into clusters, and each oracle computes a secret key and a shared public key. The public key is submitted when 1) a cluster is first formed, 2) there is a change in the cluster, and 3) the set refresh time limit is reached. Once there is a valid public key, the cluster is able to transact and verify to have a consensus by using the smart contract's shared public key.

C. SYSTEM INTERACTIONS

Before deploying smart contracts, state variables are set to what is appropriate. A chosen entity can do this through voting or a decentralized autonomous organization (DAO). Such variables could be the required stake value of oracles, the maximum number of oracles in each cluster, and the number of metrics to consider. The main contract handles interactions between all entities described in the architecture while enforcing the defined access control policies. An additional smart contract would be responsible only for handling cryptographic verification. In the following sequences, we focus on the main smart contract while the latter contract concurrently functions in the background to validate transactions.

1) Registration and Set Up

Once the contracts are added to the network, a registration phase is opened for users. Every participating oracle (regular and noncers) needs to register, as well as every HEI. Scholars, on the other hand, do not have to register. The sequence of a typical registration phase can be seen in Figure 2, and is detailed as follows:

- An oracle registers the stake amount set during the deployment phase. The SC prevents an account from registering as an oracle if it is associated with any other role on the network.
- A new cluster is generated if the oracle is the first oracle to register or if the other clusters have reached their full capacity. Otherwise, the oracle is assigned to an existing cluster. While oracles register with the appropriate stake, they get assigned to form a cluster to execute tasks.
- Once a cluster has a set amount of oracles and is ready to accept tasks, it employs a distributed key generation (DKG) protocol such as what has been detailed by Pederson [31]. The DKG protocol yields an aggregate public key with individual confidential secret keys for each oracle.
- The oracle head submits the aggregate public key, which is associated with the cluster for future transactions by the smart contract.
- During the registration phase, one of the users enlists to become a storage space (IPNS) maintainer. While fulfilling this role, the maintainer cannot assume any

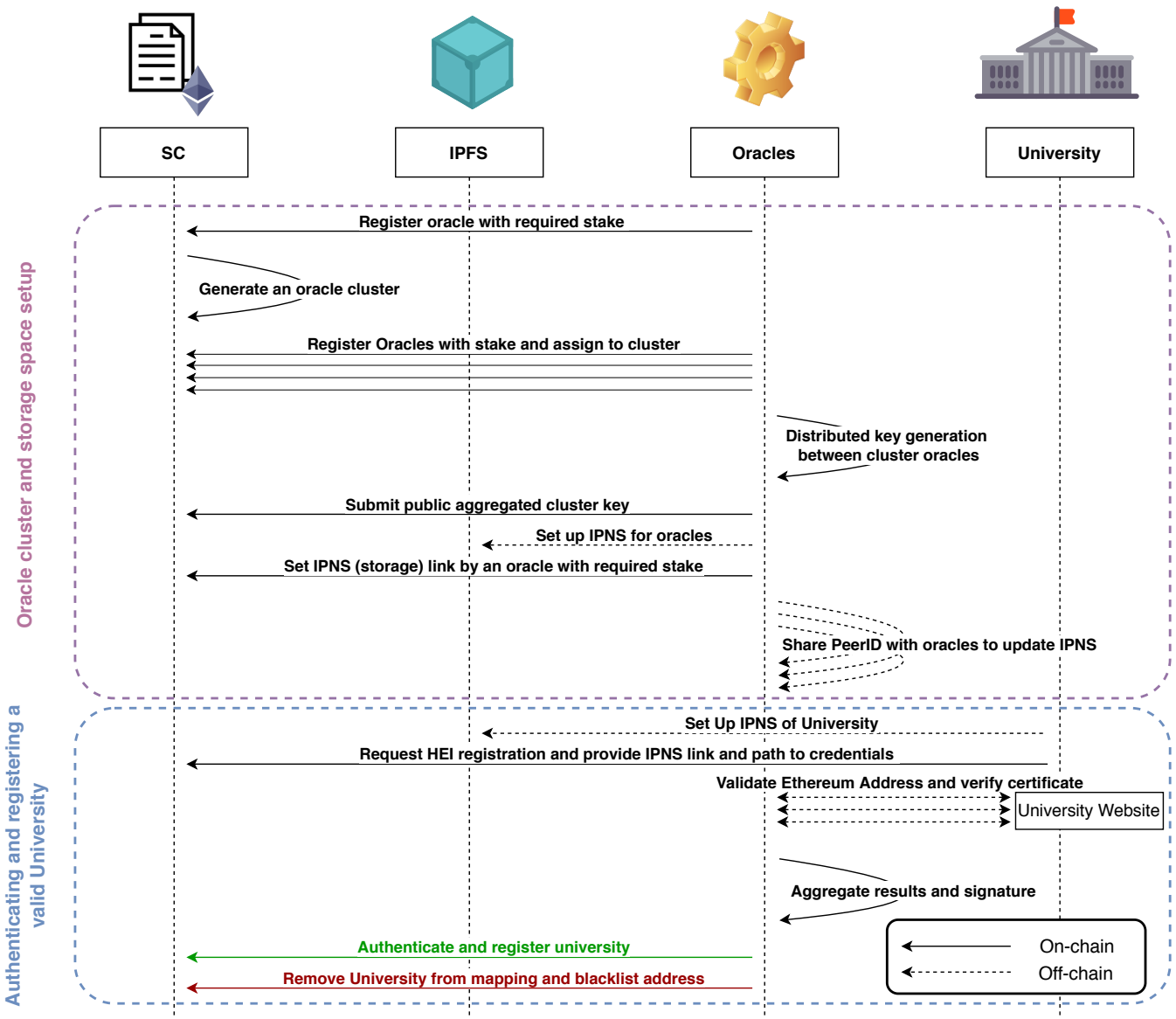

FIGURE 2. Sequence diagram of the network's registration and set-up phase, which details the generation of clusters, submission of keys, and authentication of HEIs.

other role and gains rewards for continuous cooperation. The maintainer shares the PeerID with oracle heads so that they can add data to the IPNS.

- Before an HEI registers, it sets up an IPNS with all the relevant information regarding the institution, including an anonymous list of the public keys of scholars and their departments.
- The HEI proceeds to register, providing the name, IPNS, and path to the account address on its website. The smart contract keeps a record of the HEI but does not register it yet. It then instructs the oracles to authenticate the HEI.
- Assigned oracles validate the account address's correspondence to the provided address and validate the website's authenticity through its SSL certificate. The oracles aggregate their results and signatures to be submitted to the smart contract.
- If the number of oracles confirming the account's authenticity is below a certain threshold, it is not authenticated and not registered.

2) Data Gathering

Once a few oracles, an HEI, and an IPNS maintainer have registered, the smart contract then allows rating to commence. Each HEI has an assigned oracle cluster handling its survey, while any oracle on the network can submit other data. When the survey period ends, the results are amalga-

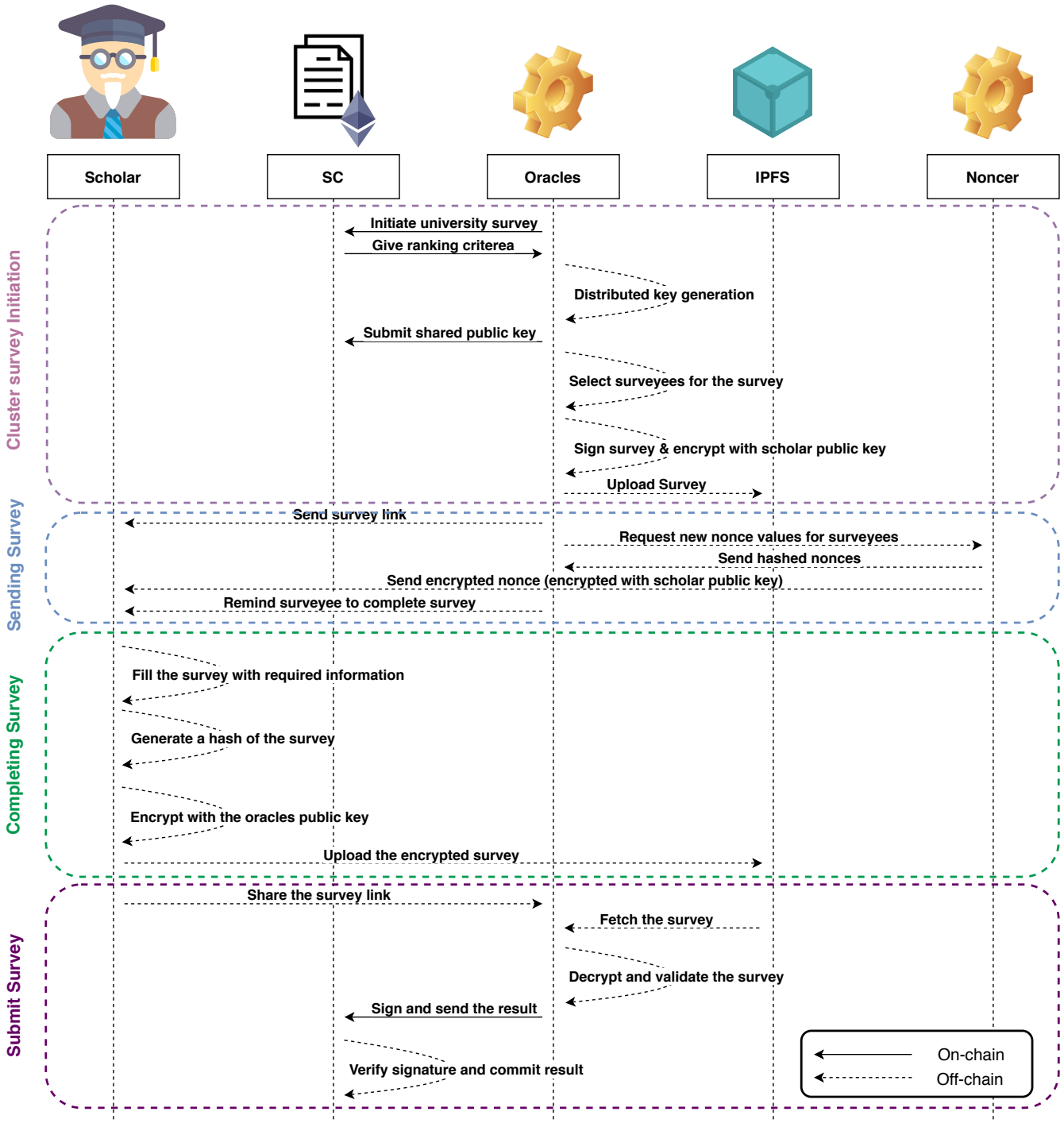

FIGURE 3. Sequence diagram of the ranking phase focuses on the survey initiation, exchange, and submission.

mated and represented in the desired form. Figure 3 clarifies the process, which occurs as follows:

- An oracle cluster requests to handle an HEI's survey, given that it satisfies the conditions for being able to execute tasks.
- The smart contract assigns the task to the cluster if it is fit and emits the type of survey that oracles have to conduct.
- If the cluster does not already have a key or has not been updated, a new one is generated. The cluster oracles execute the DKG protocol, each having a secret key and a part of the public key.
- The shared public key is then submitted to include all the current participating oracles in the cluster.

- Oracles select the surveyees, determining their public keys and the type of surveyee.
- M-of-N oracles sign the survey with their secret keys, and one oracle is assigned to encrypt it using the scholar's public key, ensuring that only the scholar receives it.
- The survey is uploaded to IPFS by the same Oracle that encrypted it. The oracle is accountable for the scholar survey.
- Oracles then send a survey link to the scholar while keeping time. The scholar is chosen from public records, and the associated public key is unknown. Scholars point to their public keys out of order and anonymously.
- An oracle from the cluster requests a set of nonces from the noncers.
- Noncers generate the nonces, hash them, and send them to the oracle cluster. Hashes are sent to the cluster to keep track of misbehavior.
- The nonce is sent securely to the scholar, which the scholar hashes and includes in the survey.
- The oracle cluster sends a reminder to fill out the survey if the scholar has not completed it yet, ensuring that the reminder is sent by a different oracle.
- Surveyees complete the survey as instructed, generate a hash from the nonce, and encrypt the shared public key with the oracle.
- Surveyees upload the survey to IPFS and share the link with the oracles.
- Oracles fetch the survey, decrypt it, and validate the provided hash and signature. Moreover, the oracles validate that the survey is filled out appropriately.
- Finally, M-of-N oracles sign the result and send the transaction to the smart contract to verify and update the state.

D. SCORE INTERPRETATION

The curated data, which constitutes the metrics and indicators to evaluate HEIs, is stored in distributed storage. Once the data is uploaded, it is maintained even if the uploader no longer hosts the data. So, the participants still have access to the data and can use the indicators to make a good evaluation.

To avoid the common issue of discrepancies in indicator values, we can standardize the indicators by applying z-transformations to generate z-scores [32]. Given the z-scores, we represent each indicator based on a categorical or visual representation for users through the DApp.

$$z_i = \frac{x_i - \bar{x}}{s} \tag{1}$$

$$Score = \frac{\sum_{i=1}^{n} z_i w_i}{\sum_{i=1}^{n} w_i} \tag{2}$$

Equation 1 represents the z-function, where x is the raw indicator value, $\bar{x}$ is the indicator mean, and s is the indicator standard deviation. The yielded z-score is utilized in Equation 2, where each indicator z-score (z_i) is multiplied by its associated weight w_i, and the result is standardized by dividing by the sum of the weights w_i. The above process provides scores for each HEI, which can be used to devise different rankings and representations.

As the literature review discusses, individual ranks are focused on simplifying an HEI's overall performance. The overall score should be the primary metric on which different representations can be built. U-Multirank utilizes a median-based approach, where HEIs are assigned to a group based on how far or close they are from the median score. THE ranking uses a percentile rank approach, whereas ARWU and QS assign a full score to the highest-scoring institute and calculate the rest of the institutions as a percentage of the highest score. In the approaches above, the ranking is relative, creating a clear distinction between institutes even if the actual differences are minimal. On the other hand, approaches like U-Multirank minimize the distinction between institutes, even if there are significant differences.

As such, the end-user should assess the discrepancies of institutes through non-complex representations of data. End-users define and select the metrics and rankings that suit them, and the system should be able to display the scores of the institutes based on the selected indicators. We can achieve this effortlessly with a weighted sum that has been standardized, as indicators can be plugged into or removed from the equation. Finally, the vast number of indicators are reduced to a single score, but the end-users would still have an absolute score that represents the performance of a set of indicators rather than a ranking relative to other institutes. In the end, the scores can be represented similarly to approaches such as U-Multirank, which designates a category for each institute based on the generated score by giving it a letter grade or a visual representation.

IV. IMPLEMENTATION AND VALIDATION

In this section, we present the algorithms implemented in our smart contracts. We validate their functionality using a variety of tests. The validation is divided into phases that we test individually and then integrate to test as a whole. We implement the smart contracts required for our approach on the Ethereum blockchain, which is a public permissionless blockchain network. We enforce our solution's conditions using smart contracts to create the desired approach. We employ the Solidity programming language for Ethereum smart contract programming. As for the development environment, we rely mainly on Hardhat for compilation, debugging, deployment, and testing. We develop an automated test that executes the default flow while also handling intentional exceptions. Smart contracts run on version 0.8.15 of Solidity, and tests are written in Javascript with the aid of the Waffle and Chai libraries. For the purpose of testing our smart contracts, we use an aggregate signature scheme

Algorithm 1: Oracle Registration

Require: value == stake, notOracle
if *Cluster reached full capacity* ***OR*** *No clusters yet* **then**
  function: $generateCluster()$
else
  Increment cluster number of oracles
end
Create $Oracle$ instance
Set $Oracle$ cluster to current cluster number
Set $Oracle$ number to order in cluster
Ensure $Oracle$ gains no rewards for previous cluster tasks
Add $Oracle$ to **registered** mapping

implementation based on the BLS signatures[2]. Aside from providing the required characteristics, we also select this option because BLS signatures (BN254 elliptic curve) are set to be a significant component of the upcoming merge.

A. SMART CONTRACT ALGORITHMS

Herein, we explain the main smart contract, which handles oracle management, data management, and HEI rating.

The sequence starts with the oracle sign-up. In this phase, each oracle executes a $registerOracle$ function. Each oracle has to send a value with the transaction equal to the required stake. Furthermore, the oracle should have been registered beforehand as something other than an oracle, a noncer, or an HEI. If requirements are satisfied, the contract checks if the last cluster reached its total capacity or if this is the first cluster. If any of the conditions above are true, the smart contract generates a new cluster, and the oracle is enrolled in it. If that is not the case, then the oracle is added to an existing cluster. After assigning the oracle to a cluster, it is initialized with the cluster's number. Finally, the oracle is added to the registered mapping to ensure no overlap in roles, as shown in Algorithm 1.

One of the oracles can also call the function $setIPNSLink$ to be the maintainer. The oracle has to provide a stake higher than an ordinary cluster oracle. The task of the maintainer is to provide a peer ID to the participating clusters. HEIs also have to register by providing the name of the HEI, an identity validation path, and an IPNS link for their curated data. An entry is created from the provided HEI information so that oracles can authenticate it, but it is not registered and cannot be evaluated. Before evaluating an HEI, each cluster must submit its shared public key. The function to submit the key can only allow the cluster head to submit.

Once the cluster completes that process, it can attempt to authenticate an HEI. The cluster validates the provided path, making sure it contains the same address and a valid corresponding certificate. M-of-N oracles in the cluster do this, then they reach a consensus and aggregate their signatures. As detailed in Algorithm 2, the cluster must provide the HEI address, the result, and its signature. Only one oracle in that specific cluster can execute the function. The smart contract also ensures that the correct oracle address is used, that the HEI is not already authenticated, and that the cluster has the right to authenticate HEIs. The next step is to verify the signature using the provided public key and the result. If it is invalid, the transaction fails. If the signed message is greater than 0, then the HEI is valid and can be registered. Otherwise, the smart contract rejects the instance and deletes its entry from the ledger. Finally, regardless of the resulting outcome, the cluster is rewarded for submitting the result.

Algorithm 2: Authenticate HEI

Input: $address$ _HEIAddress, $uint$ _result, $uint[2]$ _thresholdSignature
Require: isOracle
Require: HEI address corresponds to valid HEI && HEI not authenticated yet && Cluster size >= minimum required cluster size
Require: **function:** verifySignature(_thresholdSignature, pubkey, _result)
if *result > 0* **then**
  HEI authentication flag ← true
  emit: "HEI is registered"
else
  delete HEI instance
  emit: "HEI failed to register"
end
Increment cluster completed tasks

Authenticated HEIs are now valid for surveys, and a cluster initiates a request to handle the survey of the HEI. A valid oracle requests a survey for an authenticated HEI, and the cluster is assigned the task given that no cluster is already assigned. The oracles conduct the survey as detailed in Figure 3 off the chain. As for the on-chain transaction, it is as shown in Algorithm 3. An oracle specifies the HEI address, survey result, aggregated value, message, and corresponding signature. In this case, the signed message is the survey result link, which has the aggregated values. A score instance is created for the HEI if the signature is valid. The score structure holds the metrics and links to their data, which is respectively the submitted metric score and the submitted metric data link. Afterwards, the smart contract increments the cluster number of completed tasks to withdraw its reward. Finally, the smart contract emits an event to notify the cluster to update the metrics in the governed IPNS space.

At this point, we have data from surveys and the data submitted by the HEI. Further data needed for metrics is fetched from the aforementioned public sources by oracles or submitted by network participants. Oracles or users call the function $newDataSubmission$, which would include the data, its source, the indicator, and the corresponding

[2]BLS contract: https://github.com/thehubbleproject/hubble-contracts/blob/master/contracts/libs/BLS.sol

Algorithm 3: Submit Survey Results

1 **Input**: $address$_HEIAddress, $string$ _surveyResults, $uint$_aggregatedValues, $uint[2]$_ThreshSignature, $uint[2]$_message
2 **Require**: isOracle
3 **Require**: **function:** verifySignature(_thresholdSignature, pubkey, _result)
4 Create $Score$ Instance
5 Score metrics ← _aggregatedValues
6 Score metricData ← _surveyResults
7 Increment cluster completed tasks
8 **emit:** UpdateMetrics(_HEIAddress, assignedCluster)

HEI. Once the request is out, the HEI's assigned cluster is responsible for validating the data. As shown in Algorithm 4, the submitter address, the HEI address, the metric score and its order, the metric data, the message, and its signature are submitted by the cluster. The message to be verified is a concatenation of the data owner, HEI address, and the metric's order. Only an oracle registered with the assigned cluster can execute the function. Once the signature is verified, the metric score and metric data path are updated. As before, the smart contract increments the completed task counter to reward the oracles. Finally, the smart contract clears the mapping of the submitted data and the cluster.

Each oracle can withdraw rewards from the smart contract based on the completed tasks. The smart contract keeps track of the number of withdrawals for each oracle and its authorized rewards.

Algorithm 4: Commit HEI Data

1 **Input**: $address$ _dataOwner, $address$ _HEIAddress, $uint$_metricOrder, $uint$ _metricScore, $string$ _metricData, $uint[2]$ _message, $uint[2]$ _threshSignature
2 **Require**: isOracle
3 **Require**: assignedCluster != 0 && dataOwner assigned to this cluster
4 **Require**: **function:** verifySignature(_thresholdSignature, pubkey, _message)
5 _HEIAddress $Score$ metrics ← _metricScore
6 _HEIAddress $Score$ metricData ← _metricData
7 Increment $oracle$ indivdual task count
8 Remove data owner from mapping

B. FUNCTIONALITY CORRECTNESS VERIFICATION

We validate our proof-of-concept using the Hardhat environment, carrying out an integrated test of the default scenario of our approach. While going through the use case, we run tests on the used function to validate correct behavior for various parameters and enforce access control policies. In particular, we use the Chai library[3] for advanced assertions and exception handling and use Hardhat network helpers to manipulate time[4]. Figure 4 gives an overview of some of the handled exceptions in their respective phases.

```
Handled Exception:  Oracle prevented from registering due to insufficient stake!
Handled Exception:  Oracle prevented from registering again in the same cluster!
Handled Exception:  Oracle prevented from registering again to a new cluster!
   ✓ Register Oracles with a stake
Handled Exception:  Account is not an oracle, prevented from submitting IPNS link!
Handled Exception:  Oracle prevented from submitting IPNS link due to insufficient stake!
Handled Exception:  Oracle prevented from becoming new maintainer, maintainer already exists!
   ✓ Set the IPNS
Handled Exception:  HEI prevented from submitting registration again!
Handled Exception:  Oracle prevented from submitting an HEI request!
Handled Exception:  A submitted aggregate signature is rejected due to message inconsistency!
Handled Exception:  Oracle prevented from submitting public key as it is not oracle cluster head!
Handled Exception:  Account prevented from submitting public key as it is not oracle cluster head!
Handled Exception:  Oracle from different cluster prevented from submitting public key!
Handled Exception:  Oracle prevented from authenticating due to invalid message!
Handled Exception:  Oracle prevented from authenticating due to invalid signature!
Handled Exception:  Oracle prevented from authenticating HEI when cluster not authorized!
Handled Exception:  Oracle prevented from authenticating an invalid HEI!
Handled Exception:  Oracle prevented from authenticating due to public key timeout!
   ✓ Process HEI registration request and authenticate using oracles
Handled Exception:  Account prevented from initiating a survey, should be an oracle!
Handled Exception:  Oracle prevented from initiating a survey with an invalid cluster!
Handled Exception:  Oracle prevented from initiating a survey for an unauthenticated HEI!
Handled Exception:  Oracle prevented from initiating a survey on behalf of another cluster!
Handled Exception:  Oracle prevented from initiating the survey due to public key timeout!
Handled Exception:  Oracle prevented from initiating a survey for an HEI that has an assigned cluster!
Handled Exception:  Survey result rejected due to a non-cluster oracle call!
Handled Exception:  Survey result rejected due to invalid HEI selection!
Handled Exception:  A submitted aggregate signature is rejected due to invalid message!
Handled Exception:  A submitted aggregate signature is rejected due to invalid signature!
   ✓ Conduct survey and submit results
Handled Exception:  Account already submitted HEI data!
Handled Exception:  Prevented from submitting data for an invalid HEI!
Handled Exception:  Data commit rejected due to invalid oracle!
Handled Exception:  Data commit rejected due to providing invalid data owner!
Handled Exception:  Data commit rejected, invalid HEI!
Handled Exception:  Data commit rejected, invalid message!
Handled Exception:  Data commit rejected, invalid signature!
Handled Exception:  Data commit rejected, public key refresh required!
   ✓ Submit new HEI data by oracles
Handled Exception:  Oracle not allowed to withdraw reward, did not complete tasks!
   ✓ Withdraw oracle rewards
```

FIGURE 4. A test sample validating the correctness through exception handling.

We start with the expected functionality of this compartment. A minimum number of oracles are required to register to form a cluster. Also, a maintainer is required for the IPNS used by the participants. Finally, HEIs can invoke registration requests, which the oracles will handle. We depict some of the most important exceptions to handle and required assertions. We refer to assertions as (A) and handled exception as (HE).

1) Oracle registration

Each oracle registers with the specified stake, determined at smart contract deployment. Each oracle is automatically assigned, and thereafter confined, to a cluster. In the test, we registered 11 oracles successfully out of 15 registration attempts. The four invalid registration attempts were expected and handled accordingly. The handled exceptions and assertions of right behavior are as follows:

- A: The first oracle is assigned to the first newly generated cluster.
- HE: An oracle attempts to register with a stake lower than expected. As such, it was rejected and not registered.
- HE: A registered oracle attempts to register again but is prevented from doing so.
- A: The 11th oracle is assigned to a newly generated cluster, as the maximum threshold has been set to 10 in this test.

[3]https://hardhat.org/hardhat-chai-matchers/docs/overview
[4]https://hardhat.org/hardhat-network-helpers/docs/overview

- HE: A previous oracle attempts to register for the newly generated cluster but is prevented from doing so.

2) IPNS setup

One of the registered oracles can be the maintainer of the IPNS link. When registering as the maintainer, the stake is x times greater than the original. For this test, we assume x is 5, and 4 attempts are taken for an oracle to register, given the following cases:

- HE: An account is prevented from being the maintainer for not being a registered oracle.
- HE: An oracle is prevented from being a maintainer due to an invalid stake amount.
- A: A registered oracle with a valid stake amount registers as a maintainer.
- HE: An oracle is prevented from being a maintainer, as there is already a valid maintainer.

3) HEI registration

For an HEI to be ranked, it must register and authenticate. Each HEI submits its official name, a domain path with its address, and an IPNS path. The smart contract creates an entry for the HEI, which the oracles use to authenticate the HEI. Furthermore, while this process is ongoing, the HEI is prevented from initiating new requests to prevent service denial attacks by malicious users. We start the test with an accepted registration request, followed by two failed attempts:

- HE: A previously registered HEI is prevented from submitting new requests following the first accepted request.
- HE: An oracle attempts to pose as an HEI and is prevented from submitting the request.

We also submitted a request for a different HEI. Once an HEI has requested registration, clusters can submit their authentication results. The first valid cluster with valid authentication is assigned to the HEI.

4) Key submission

Most of this compartment is done off-chain, with oracles generating a public key and their secret keys. We also simulate an invalid cluster, where one of the oracles signs an invalid message. In the test, the following assertions and exception handling occur:

- A: The generated signature from the valid cluster is verified and considered valid.
- A: The generated signature from the invalid cluster (with a faulty message) fails the verification.
- HE: An oracle that is not the head of the cluster attempts to submit the shared public key and is prevented.
- HE: An account not part of the cluster attempts to submit the key and is prevented.
- HE: An oracle from a different cluster attempts to submit a key for a different cluster and is prevented.

5) Authenticating HEI

HEIs that are viable for authentication can be authenticated by any cluster that submits a valid request. The cluster must have the minimum number of oracles defined upon deployment. Furthermore, they should submit a valid threshold signature. We handle exception cases as follows:

- HE: An oracle from a valid cluster submits the authentication request but is rejected due to an invalid message.
- HE: An oracle from a valid cluster submits the authentication request, but it is rejected due to an invalid threshold signature.
- HE: An oracle from a cluster with a low number of oracles submits an authentication request and is rejected for not meeting the minimum number of oracles.
- HE: An oracle from a valid cluster submits an authentication request for an invalid HEI and is rejected.
- HE: An oracle from a valid cluster with signature details submits an authentication request but is rejected due to the timeout of the public key of the cluster.

After updating the smart contract's shared public key, the last cluster can verify the HEI.

6) Conducting surveys

A valid cluster initiates a survey, and the cluster is assigned to the HEI if successful. The assigned cluster handles any further requests for the HEI. The initiation does not require the signature of the cluster. Moreover, the key timeout is also checked before accepting the request. We test the surveying of an HEI and handle exceptions as follows:

- HE: An account is prevented from submitting a survey request if it is not a member of the cluster.
- HE: An oracle that is part of an invalid cluster is prevented from initiating the survey request.
- HE: An oracle is prevented from initiating a survey for an unauthenticated HEI.
- HE: An oracle from a different cluster is prevented from initiating a request on the cluster's behalf.
- HE: An oracle is prevented from initiating a survey request since the public key needs to be resubmitted.
- A: The survey is initiated by a valid oracle from a valid cluster.
- HE: An oracle is prevented from initiating a survey for an HEI that has been assigned a cluster.

7) Submitting survey result

When submitting signed results, we use an N-to-N threshold signature relationship in the tests. To ensure correct functionality, we simulate an oracle that generates a new pair and attempts to join the cluster consensus. We handle the cases as follows:

- A: A signature generated by Oracles is valid (off-chain).
- A: A generated signature by oracles, including an oracle with a new key pair, is invalid (off-chain).
- HE: The survey result is rejected due to an oracle from a non-assigned cluster being the submitter.

- HE: The survey result is rejected due to selecting an invalid HEI.
- HE: The survey result is rejected due to an invalid signed message.
- HE: The survey result is rejected due to an invalid threshold signature.
- A: The survey result submitted by a valid assigned oracle with the appropriate parameters.

8) Submitting new HEI data

When an HEI is authenticated, data regarding the HEI can be submitted by participants on the network. Such data is used to bolster existing data and provide a separate set of information for reference. The submitter invokes a request to submit new data for the oracles to validate. We validate the function as follows:

- A: A valid account invokes a new data submission request for a valid HEI.
- HE: The account attempts to invoke a new request but is prevented from doing so by an active request.
- HE: An account is prevented from submitting a request for an invalid HEI.

9) Committing HEI data

An oracle from the assigned cluster has to submit a transaction with the signature of a set number from the cluster. The transaction would be a validation of the submitted data, which is committed to the blockchain through the transaction. We test the function as follows:

- A: An oracles-generated signature is valid (off-chain).
- A: A generated signature by oracles, including an oracle with a new key pair, is invalid (off-chain).
- HE: An oracle that is not part of the assigned cluster is prevented from committing the data.
- HE: An oracle attempted to commit the data by providing a different data owner but was prevented.
- HE: Data commit rejected due to an invalidly signed message.
- HE: Data commit rejected due to an invalid threshold signature.
- HE: Data commit rejected. The cluster must submit a new public key.

V. EVALUATION AND ANALYSIS

In this section, we present the cost, performance, and security analyses to evaluate the proposed solution.

A. GAS COST ANALYSIS

We build our proposed solution on the Ethereum blockchain, which offers advanced smart contract capabilities. The solution is not constrained to Ethereum and can be migrated to other networks that support smart contracts. Costs are enumerated in terms of gas, the quantifiable unit that estimates the computational consumption on a machine [33]. The monetary costs depend on the price of a gas unit and the chosen currency's value. In our case, the average gas price was 21 Gwei. While the price of ether keeps fluctuating, the amount of gas is consistent for operations. As such, gas is a more accurate representation of the cost efficiency of the solution.

We generate the costs of the methods executed in our validation and described in our algorithm sections, as seen in Table 1. The average of the function costs was taken based on the number of times it was called. As can be seen, there is a drastic cost difference between the methods due to signature verification. All the methods that internally call the verification function start from $200,000$ gas, namely authenticateHEI, commitHEIData and submitSurveyResult. Such costs are necessary to maintain security seamlessly between off-chain and on-chain transactions. Simple request functions that do not require substantial oracle participation come with modest gas costs, which are around 50, 60, and 80 thousand gas. Not only does this approach require only one transaction, facilitating lower traffic, but using threshold signatures also decreases costs considerably due to the low number of calls. Regardless of the number of oracles, initiateSurvey and submitSurveyResult only have to be called once to reach a consensus. If the smart contract handles the consensus, the function would be called N times, which is equal to the number of oracles N required for each survey initiation and result submission.

B. THROUGHPUT AND LATENCY ANALYSIS

We examine the throughput of transactions because our solution focuses on managing and curating data through smart contract governance. Although the contracts are developed using the Solidity programming language, they can be migrated to other ecosystems. For the sake of our implementation, we inspect Ethereum and Polygon, which rely on Solidity smart contracts. The PoW Ethereum network has an average block mining time of 13 seconds, and the maximum block size is 30 million gas. As for Polygon, the average time is 2 seconds, and the maximum block size is also 30 million. The transaction latency depends on the priority fee and current network traffic. A high priority fee with low traffic provides the lowest latency, while a low priority fee with high traffic yields the worst latency. Simulating for the average block time gives us the provided transaction throughput shown in Table 2. We define the throughput as the number of actions per second and generate the throughput per function. We run our default test on Hardhat's development network with instant mining; the test itself takes 7 seconds. Rerunning the same test with a mining interval of 6.5 seconds increases the test time to 15 seconds. The latency depends on when the transaction is executed compared to when a block is mined. In the typical case, the latency of the transaction would be the average block time in a non-congested network.

TABLE 1. Gas cost of implemented functions on Ethereum and Polygon

Method	Average Gas Cost	Ether	Ethereum-USD (Average)	Matic	Matic-USD(Average)
authenticateHEI	205129	0.0043	7.52	0.0065	0.0060
commitHEIData	223133	0.0047	8.18	0.0071	0.0065
initiateSurvey	67454	0.0014	2.47	0.0021	0.0020
newDataSubmission	50897	0.0011	1.86	0.0016	0.0015
registerHEI	146639	0.0031	5.37	0.0046	0.0043
registerOracle	134936	0.0028	4.94	0.0043	0.0040
setIPNSLink	118028	0.0025	4.32	0.0037	0.0035
submitPubKey	78463	0.0016	2.87	0.0025	0.0023
submitSurveyResult	298019	0.0063	10.92	0.0094	0.0087
Deployment					
Oracles ( )	2651451	0.0557	110.52	0.0838	0.0776

TABLE 2. Throughput of the implemented smart contract

Method	Total Gas Cost	Ethereum Throughput	Polygon Throughput
authenticateHEI	205129	11.2	73.1
commitHEIData	223133	10.3	67.2
initiateSurvey	67454	34.2	222.4
newDataSubmission	50897	45.3	294.7
registerHEI	146639	15.7	102.3
registerOracle	674680	3.4	22.2
setIPNSLink	118028	19.6	127.1
submitPubKey	78463	29.4	191.2
submitSurveyResult	298019	7.7	50.3
Deployment	2651451	0.9	5.7

TABLE 3. Security vulnerability analysis of the smart contracts

Impact \ Confidence	Low	Medium	High
Low	0	3	0
Medium	0	4	0
High	0	0	0
Informational	0	7	32
Optimization	0	0	20

C. SECURITY REQUIREMENTS AND VULNERABILITY ANALYSIS

We test our developed smart contracts using a vulnerability analysis tool called Slither. Upon conducting the analysis, we unveiled seven concerns, as shown in Table 3. Three are reported as low impact with medium confidence, while four vulnerabilities are categorized as having medium impact and confidence. The low impact concerns pertain to using timestamps in comparisons. However, we validate correct usage and see no threat to the developed contracts. As for medium impact vulnerabilities, the four concerns are for uninitialized variables, which are our structures; however, these were false positives, and we ensured correct instantiation of the structure variables. The remaining messages of the report were informational, and regarding code optimization, no real threats were detected through the static vulnerability analysis of the contract.

In the following, we present the critical required security aspects of the proposed solution and highlight specific phases or compartments of concern for each aspect:

- **Anonymity**: As the system relies on data submitted by different entities, accountability may be a deterrent to honest and unbiased data. As such, in our solution, we preserve the anonymity of the surveyees and data owners using the Ethereum blockchain, which ensures pseudo-anonymity. We do not demand that surveyees be on the blockchain, and the survey process off-chain does not associate the survey with the identity of the surveyees. During selection, only the key pair is selected to encrypt data without associating it with an identity. The smart contract sends an event to all HEI participants, allowing the keypair owner to decrypt the survey on IPFS. Once a surveyee completes a survey, the file is encrypted with the oracle public key and its associated nonce to ensure the surveyee is valid without breaching anonymity. During submission, oracles submit the composite survey results on-chain while surveys are kept off-chain. The surveys also do not contain PII, so anonymity is preserved both on-chain and off-chain.
- **Availability**: The approach's availability is a significant advantage over traditional server-based approaches. Decentralization is at the approach's core, which we highlight in the different utilized components. On the network level, the blockchain is utilized for transaction execution and on-chain storage. As for storage, the IPFS is leveraged; it is a well-established distributed storage peer network with many participants, making it very difficult to disrupt its service. Furthermore, governance is also decentralized through automated policies executed by smart contracts, which are maintained by the decentralized blockchain nodes, offering redundancy.
  As for the oracle layer, we utilize a similar concept where decentralized oracle clusters cooperate to validate

the content and reach a consensus. Since the oracles use threshold encryption to reach consensus, the cluster can continue operation with M of N oracles in the cluster. As such, attacking and disrupting the service of an oracle is not enough; the number of oracles needs to go below the minimum threshold to stop cluster operations. Furthermore, the smart contract can reassign a new cluster to the task, so our approach offers high availability even at the oracle level.

- **Integrity**: Lack of transparency and data integrity are the main concerns in current ranking systems. Integrity on the blockchain is maintained through the hashing mechanism and digital signatures, which are generated from participants' keypairs. Smart contracts validate off-chain data managed by oracles, and the oracles also verify surveyee data. Any interaction between an oracle and a surveyee is encrypted using asymmetric encryption. Moreover, a nonce is included in the encrypted data to ensure a valid surveyee submits data without breaching anonymity. Noncers generate the nonce, and the hash of the nonce is shared with oracles to validate them. Once data is submitted by the cluster and validated by the smart contract, it is immutably recorded on the blockchain, preserving the data's integrity.
- **Transparency**: The blockchain provides an immutable audit trail of transactions saved in blocks. Current methods explain their methodology but do not provide tangible insight into its operation, offering only a vague description. In our approach, the composite indicators, the participating oracles and the process of surveying scholars are transparent through the blockchain. Furthermore, off-chain, the data is stored on persistent distributed storage, which is utilized for details of surveys and HEI data. Finally, we manage clusters through smart contracts to facilitate the blockchain's intrinsic transparency. The oracles can submit invalid data and misbehave, but the misbehavior is then handled by the smart contract, where an oracle can be penalized or disqualified. Transparency is leveraged here by letting misbehavior occur before taking action so that there is an audit trail that serves as proof of misbehavior and justifies retaliative action.

VI. CONCLUSION

In this paper, we have proposed a blockchain-based solution to enhance the academic ranking systems in a manner that is decentralized, transparent, traceable, auditable, and trustworthy. We presented in detail the proposed system architecture and sequence of interactions between the academic entities involved in the ranking systems. We developed smart contracts to enforce automated and decentralized governance. We integrated the Ethereum blockchain with IPFS and oracles to handle external data submission and scholar surveying. Trust is enforced off-chain through consensus built by threshold encryption, which is done in a scalable manner by simplified state channels. We presented algorithms, validated and tested our contracts, and analyzed the proposed solution through cost, throughput and latency, and security parameters. The evaluation results revealed that the proposed solution is affordable and secure against attacks. In future work, we will thoroughly test the prototype and enhance it to be deployment-ready. Additionally, we will determine the most desirable representation methods and incorporate them into the DApp, as well as test this work on Ethereum 2.0.

VII. ACKNOWLEDGEMENT

This publication is based upon work supported by the Khalifa University of Science and Technology under Award No. CIRA-2019-001.

REFERENCES

[1] F. Selten, C. Neylon, C.-K. Huang, and P. Groth, "A longitudinal analysis of university rankings," Quantitative Science Studies, vol. 1, no. 3, pp. 1109–1135, 2020.
[2] K. Soh, "What the Overall doesn't tell about world university rankings: examples from ARWU, QSWUR, and THEWUR in 2013," Journal of Higher Education Policy and Management, vol. 37, no. 3, pp. 295–307, 2015.
[3] F. Anowar, M. A. Helal, S. Afroj, S. Sultana, F. Sarker, and K. A. Mamun, "A critical review on world university ranking in terms of top four ranking systems," New trends in networking, computing, e-learning, systems sciences, and engineering, pp. 559–566, 2015.
[4] M. A. Fauzi, C. N.-L. Tan, M. Daud, and M. M. N. Awalludin, "University rankings: A review of methodological flaws," Issues in Educational Research, vol. 30, no. 1, pp. 79–96, 2020.
[5] E. Hazelkorn, "Measuring world-class excellence and the global obsession with rankings," in Handbook on globalization and higher education. Edward Elgar Publishing, 2011.
[6] S. Marginson, "University rankings and social science," European journal of education, vol. 49, no. 1, pp. 45–59, 2014.
[7] M. M. Vernon, E. A. Balas, and S. Momani, "Are university rankings useful to improve research? A systematic review," PloS one, vol. 13, no. 3, p. e0193762, 2018.
[8] I. Chirikov, "Does conflict of interest distort global university rankings?" Higher Education, pp. 1–18, 2022.
[9] B. Kehm, "Global university rankings: impacts and applications," Gaming the Metrics, p. 93, 2020.
[10] A. Calderon, "New rankings results show how some are gaming the system," University World News, vol. 12, 2020.
[11] K. Soh, "Misleading university rankings: Cause and cure for discrepancies between nominal and attained weights," Journal of Higher Education Policy and Management, vol. 35, no. 2, pp. 206–214, 2013.
[12] K. Soh, "Problems of indicator weights and multicolinearity in world university rankings: comparisons of three systems." Higher Education Review, vol. 46, no. 2, 2014.
[13] M. P. Çakır, C. Acartürk, O. Alaşehir, and C. Çilingir, "A comparative analysis of global and national university ranking systems," Scientometrics, vol. 103, no. 3, pp. 813–848, 2015.
[14] J. Berbegal-Mirabent and D. E. Ribeiro-Soriano, "Behind league tables and ranking systems: a critical perspective of how university quality is measured," Journal of Service Theory and Practice, 2015.
[15] I. F. Aguillo, "Ranking web of universities." [Online]. Available: https://www.webometrics.info/en/Methodology
[16] L. Waltman, C. Calero-Medina, J. Kosten, E. C. Noyons, R. J. Tijssen, N. J. van Eck, T. N. van Leeuwen, A. F. van Raan, M. S. Visser, and P. Wouters, "The Leiden Ranking 2011/2012: Data collection, indicators, and interpretation," Journal of the American society for information science and technology, vol. 63, no. 12, pp. 2419–2432, 2012.
[17] G. Chen, B. Xu, M. Lu, and N.-S. Chen, "Exploring blockchain technology and its potential applications for education," Smart Learning Environments, vol. 5, no. 1, pp. 1–10, 2018.
[18] L. Liu, M. Han, Y. Zhou, R. M. Parizi, and M. Korayem, "Blockchain-based certification for education, employment, and skill with incentive mechanism," in Blockchain cybersecurity, trust and privacy. Springer, 2020, pp. 269–290.

[19] IREG Inventory on International Rankings, available Online: https://ireg-observatory.org/en/wp-content/uploads/2021/03/IREG-Inventory-2021-final-report-2021-03-19.pdf, accessed: 2022-11-16.
[20] E. Gadd, R. Holmes, and J. Shearer, "Developing a Method for Evaluating Global University Rankings," Scholarly Assessment Reports, vol. 3, no. 1, 2021.
[21] L. Liyuan, H. Meng, Z. Yiyun, and P. Reza, "Eˆ 2 C-Chain: a two-stage incentive education employment and skill certification blockchain," in 2019 IEEE International Conference on Blockchain (Blockchain). IEEE, 2019, pp. 140–147.
[22] A. Garg, P. Kumar, M. Madhukar, O. Loyola-González, M. Kumar et al., "Blockchain-based online education content ranking," Education and information technologies, vol. 27, no. 4, pp. 4793–4815, 2022.
[23] X. Yang, Y. Zhang, S. Wang, B. Yu, F. Li, Y. Li, and W. Yan, "LedgerDB: a centralized ledger database for universal audit and verification," Proceedings of the VLDB Endowment, vol. 13, no. 12, pp. 3138–3151, 2020.
[24] I. Homoliak and P. Szalachowski, "Aquareum: A centralized ledger enhanced with blockchain and trusted computing," arXiv preprint arXiv:2005.13339, 2020.
[25] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich et al., "Hyperledger fabric: a distributed operating system for permissioned blockchains," in Proceedings of the thirteenth EuroSys conference, 2018, pp. 1–15.
[26] M. Saisana, B. d'Hombres, and A. Saltelli, "Rickety numbers: Volatility of university rankings and policy implications," Research policy, vol. 40, no. 1, pp. 165–177, 2011.
[27] F. A. Van Vught and F. Ziegele, Multidimensional ranking: The design and development of U-Multirank. Springer Science & Business Media, 2012, vol. 37.
[28] G. Kováts, "New Rankings on the Scene: The U21 Ranking of National Higher Education Systems and U-Multirank," in The European higher education area. Springer, Cham, 2015, pp. 293–311.
[29] N. Szabo, Smart Contracts, Phonetic sciences, available Online: https://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best.vwh.net/smart.contracts.html, accessed: 2022-11-16.
[30] L. D. Negka and G. P. Spathoulas, "Blockchain state channels: A state of the art," IEEE Access, 2021.
[31] T. P. Pedersen, "A threshold cryptosystem without a trusted party," in Workshop on the Theory and Application of of Cryptographic Techniques. Springer, 1991, pp. 522–526.
[32] K. Soh, "The seven deadly sins of world university ranking: A summary from several papers," Journal of Higher Education Policy and Management, vol. 39, no. 1, pp. 104–115, 2017.
[33] G. Wood, "ETHEREUM: A Secure Decentralized Generalised Transaction Ledger," Aug 2022. [Online]. Available: https://ethereum.github.io/yellowpaper/paper.pdf