USING BLOCKCHAIN TO ENSURE REPUTATION CREDIBILITY IN DECENTRALIZED REVIEW MANAGEMENT

Zachary James Zaccagni

Dissertation Prepared for the Degree of

DOCTOR OF PHILOSOPHY

UNIVERSITY OF NORTH TEXAS

December 2023

APPROVED:

Kirill Morozov, Major Professor
Kamesh Namuduri, Co-Major Professor
Bill Buckles, Committee Member
Cihan Tunc, Committee Member
Gergely Záruba, Chair of the Department of Computer Science and Engineering
Paul S. Krueger, Dean of the College of Engineering
Victor Prybutok, Dean of the Toulouse Graduate School

Zaccagni, Zachary James. *Using Blockchain to Ensure Reputation Credibility in Decentralized Review Management.* Doctor of Philosophy (Computer Science and Engineering). December 2023, 125 pp., 2 tables, 23 figures, references, 76 titles.

In recent years, there have been incidents which decreased people's trust in some organizations and authorities responsible for ratings and accreditation. For a few prominent examples, there was a security breach at Equifax (2017), misconduct was found in the Standard & Poor's Ratings Services (2015), and the Accrediting Council for Independent Colleges and Schools (2022) validated some of the low-performing schools as delivering higher standards than they actually were. A natural solution to these types of issues is to decentralize the relevant trust management processes using blockchain technologies. The research problems which are tackled in this thesis consider the issue of trust in reputation for assessment and review credibility at different angles, in the context of blockchain applications.

ACKNOWLEDGMENTS

I would like to express my gratitude to my major professor, Kirill Morozov, for his guidance, advice and sharing of his expertise. His support helped hone my scientific writing and focus, while fundamentally transforming who I am as a researcher at the doctoral level. A sincere thanks goes to my master's thesis advisor and co-major professor, Kamesh Namuduri, whose continued confidence and support has always encouraged me to expand beyond my knowledge in computer science, connecting me with UNT to pursue this degree.

Thanks are also due to Professor Ram Dantu for accepting me into his lab and funding the first two years of my program. I would also like to extend my gratitude to the other members of my committee, Professor Bill Buckles and Professor Cihan Tunc.

Also, I would like to express my sincere gratitude to NSWC Crane and my colleagues who have funded me through a PhD fellowship program to finish this degree.

With great love and gratitude, I would like to say a significant thank you to all my family and friends, especially my fantastic wife and brilliant children, whose continued encouragement and sacrifices have enabled me to finish this degree. To the Moon, Mars, Meow Wolf, and back we have been to make this happen. Thank you for entertaining the probabilistic models and maths with me. Thank you for listening to me bounce my research ideas and concepts repeatedly over the years.

Thank you to my wonderful mom, who had given me the tools and opportunities for learning while growing up, and for buying me my first computer. And thanks to my parents-in-law who have always encouraged and supported me, especially in my pursuit of this degree.

My thanks to IEEE and the Association for Computing Machinery for allowing me to reproduce the following published articles in my dissertation:

- Zachary Zaccagni, Aditya Paul, and Ram Dantu, "Micro-accreditation for Matching Employer e-Hire Needs," 2019 IEEE International Conference on Blockchain (Blockchain), 2019, pp. 347–352. This material is located in Chapters 1 and 3, and sections 6.1.1 and 6.2.3.

- Zachary Zaccagni, Ram Dantu, and Kirill Morozov, "Maintaining Review Credibility using NLP, Reputation, and Blockchain," 2022 IEEE 4th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA), 2022, pp. 58–66. This material is located in Chapters 1 and 4, and sections 6.1.2 and 6.2.1.
- Zachary Zaccagni, Ram Dantu, and Kirill Morozov, "Proof of Review - Trust Me, It's Been Reviewed," 2023 5th Blockchain and Internet of Things Conference (BIOTC23), 2023 (in press). This material is reproduced with permission from the Association for Computing Machinery, and is located in Chapters 1 and 5, and sections 6.1.3 and 6.2.2.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In recent years, there have been incidents which decreased people's trust in some organizations and authorities responsible for ratings and accreditation. For example, there was a security breach at Equifax in 2017 [**25**], where 148 million records were leaked. Hackers stole personal information of users, such as names, social security numbers, birthdays, license numbers, addresses, credit card data, and other information. Another example was a misconduct found in the Standard & Poor's Ratings Services in 2015, where this company misrepresented methodology in their credit ratings process in order to raise their own financial interest above that of their investors. Most recently, the US Department of Education stripped the recognition as an accreditation agency from the Accrediting Council for Independent Colleges and Schools (ACICS) in 2022 [**64**]. This agency had been validating some of the worst low-performing for-profit schools as being high quality and crediting them as performing at higher standards than they actually were. This led to a higher-than-average loan default rates for students (about 20%) [**43**], and eventually the closing of some of the schools, leaving some students without a degree and in financial hardship. Of course, in addition, the trust in the education quality and degrees of the students who graduated from these schools came into question [**10**].

The above are only a few prominent examples of recent security incidents and breaches. This motivated us to investigate ask the following problems related to trust and reputation management for accreditation and assessments. How can we trust courses in one college to provide a student with the type and level of knowledge needed in the workplace? How can we accurately compare schools and the students graduating from them? Can we increase confidence in online reviews and trust that they reflect a credible assessment? How can we trust that this data is secured against unauthorized modification and other malicious actions?

A natural solution is to decentralize the relevant trust management processes using

blockchain technologies. Decentralization removes a single point of failure by bringing a multiple participants/stakeholders into play. In particular, a blockchain lends itself well to solving the above-mentioned issues in trust as a well-established technology for decentralization.

With these considerations in mind, we first directed our focus on accreditation for workforce development, specifically in micro-accreditation. We explored ways to help match employers' knowledge requirements with students' knowledge earned inside and outside of classes. Currently, knowing what a person has specifically learned is mostly from assumption about a course in school. How well they learned certain topics can only be gleaned from the grade they received on their transcript. For example, if one sees a class in e-commerce security, one may assume a student learned about threats, vulnerabilities, and solutions for protecting an e-commerce site from malicious players. Of course, it is possible to access to the specific knowledge/skills learned in the class by inspecting the course syllabus. However, the actual presented material evolves over time, sometimes very significantly, which may not be fully reflected in the syllabus' updates. Additionally, a student or a potential employer may not be able to adequately assess how well the student understood each particular topic in a class. The final grade does not permit such the fine-grained assessment. Currently, the only effective way to achieve the above is to invest some resources (time and money) into testing a potential hire on each and every topic that is important for the job in question. This evaluation process may be resource-consuming, prone to evaluation bias and human errors. Also, it does no scale well, meaning that say a 100 applicants will require a 100 evaluations. This begs the question whether we can shift the load on the applicant's side, seeing that the testing is routinely performed in the educational institutions anyways.

Since a blockchain is a standard way to decentralize processes, our research focused on using that to assure credentials and track student records. This decentralized approach proposes using the micro-accreditation of topics from the CAE framework [**31**]. We define a course by breaking it down to topics, knowledge units, and associated tasks. After the course is created, students enroll in classes where the knowledge and topics are explicitly

defined. What was learned in the individual topics can be discerned once the course is completed. This approach helps establish immutable proof of academic credentials and what was specifically learned. For each student, this data is stored on the blockchain or at least a location pointing to where it's stored off-chain (in a database not on the blockchain). An employer can disregard the course grade if they can see a job's specific topic requirements are met satisfactorily. Meaning, a student may have received a lower overall grade for a course, but still meets (or exceeds) the proven expertise in topics for a job.

We also introduce a blockchain-based rigor score assessment. Our micro-accreditation framework required an introduction of a peer-review system for determining a "rigor" of a course components (knowledge units). Rigor is the level of difficulty some problem is in regards to the student's expected level of understanding and knowledge. If it is more difficult and challenging, it has greater rigor. Applying rigor to topics in a course, we can better understand what a student has learned. Additionally, we can better compare what is learned from one institution to another. Peer-reviewed rigor scoring is very important for the accuracy of evaluating a student's knowledge. One example would be comparing a Calculus I class offered at a top national university versus that at an average regional university. We aim at soliciting multiple peer reviews and mitigating the effects or bias and human errors.

Assigning reputation to peers is a natural method to ensure correctness of the assessments. By doing so, those peers with a greater reputation would have a greater influence on the overall rigor assessment values. Since we needed to trust that the peer reviews were honest, we needed a completely new automated reputation system. This led to exploring decentralized reputation management. The reputation of the peers providing rigor scores needs to come into the calculation for an overall rigor for a course, its topics, and its tasks. Meaning, those with a higher reputation have more influence on the total score. How is a peer's reputation determined, gained, or lost? With that, how can we trust the reputation to be credible and not manipulated by malicious players and actions? With this, we needed to investigate and develop a system that included both reputation and evaluation systems. Moreover, the peers would need to come to a consensus on the scores for knowledge units. In

a blockchain, consensus typically means that a majority agree on something. In our case, this also means they agree on the evaluation of the peer scores. We figured we would eventually need a new consensus protocol to ensure a trust in that reputation system (discussed later in Problem 3), but focused on answering the former question first. In this implementation, we focused on a different scenario, decentralized marketplaces. We use a Proof-of-Stake based Algorand system as a base of our implementation, since this system is open-source, and it has a rich community support. It also allows for easy modifications, such as adding extra components. With this, the proposed approach to this new problem can also be fitted to the micro-accreditation, which was described above. Additionally, by using PoS, we did not have to worry about gas costs (for completing transactions) and other computational pains associated with a Proof of Work platform (PoW), and just focus on solving our problem.

The next important problem was to ensure that peer's reputation is assigned correctly. In turn, this will provide the review credibility. This led us to introducing a new consensus protocol, which we call a "Proof of Review" (PoRev). Here, providing congruent reviews earns a reputation for parties, which in turn serves as an asset ensuring their higher weight in the system. In that way, PoRev borrows from the earlier concepts of Proof-of-Stake and Proof-of-Reputation. It is worth noting that this consensus mechanism can be applied beyond reviews and the peer-reviewed assessments, although in this thesis, we are focusing on these application. Our proposed architecture includes a reputation system managed by an automated evaluation system and the minimum-reputation and blacklisting components used to protect the system from the malicious players. This system then provides a trust in evaluations (review/assessment credibility) and those who provide them (reputation credibility) using a blockchain. With this, we can trust the rigor scores to reflect an accurate value in various platforms such as micro-accreditation and reviews of products and services. Additionally, we can use this concept for other types of applications with similar needs of trust in assessments and the players providing them, including but not limited to: sensor arrays, autonomous caravanning cars, and marketplaces.

1.1. Research Focus of This Work

Let us now go into specifics of the research problems which are tackled in this thesis. They all investigate the issues of trust and reputation for assessment and review credibility at different angles, in the context of blockchain applications.

1.1.1. Problem 1: Accreditation for Workforce Development

First, we explore the issue of trust in education systems to provide knowledge and skills needed for the workforce. Currently, there are several related problems, which concern, e.g., existing hiring methods and with students transfer support. In particular, hiring is multi-stage process, which involves recruitment processes, rounds of interviews, and the related paperwork. Much of this process is performed by human resources or it is outsourced to third parties who must be trusted. Much of the complexity and confusion in finding the right candidates arise from these parties being too removed from the actual work for which an applicant is being hired. Currently, this hiring process usually lacks regular, organized, and established definitions of knowledge and skills, encompassing both academic aspects and the workforce expectations.

Generally, it is difficult to locate information on new graduates to deduce their potential performance in the new workplace. Most information about them is contained in transcripts, which are difficult to interpret correctly from only course grades and the school name. Moreover, this information is not normalized from one institution to another, and it rarely reflects what specific knowledge was learned. For this reason, comparing courses taught in different schools is very challenging. Accuracy is also an issue, since the skills required for a career may not be matched to the courses precisely, as different skills are taught in different courses at different schools.

Furthermore, it is difficult to discern if any bias existed during grading. A student's grade for an assignment or overall course could have been influenced by bias (e.g., how good was the relationship between teacher and student), diligence of the grader, and a level of oversight provided by the instructor. Finally, making mistakes in hiring may be both risky

and expensive. For recruiting and replacing a bad hire, costs can be up to $240,000, see, e.g., [**23**, **24**].

Due to the above-listed reasons, it is of tantamount importance to create a system that efficiently and accurately reduces the risk involved in the hiring process through an effective matching algorithm. In addition, this system must normalize grades as well as the rigor of an institution, while eliminating any middlemen involved in the process. Since the rigor is peer-reviewed, it reflects the most accurate standard in the score determination. In general, combining rigor and knowledge topics creates an effective solution for comparing courses from different academic institutions for either school transfers or job placements.

Our solution was to create a decentralized system. The student's records are stored in an Ethereum-based blockchain network, along with a decentralized course storage system. Each course has a required set of Knowledge Units (KU) and micro-accredited topics mapped to assignments. The KU topics are defined by the Center of Academic Excellence (CAE) [**31**]. Each assignment element will be tagged with both a KU topic and a rigor score, determined by a peer-review process. Rigor is defined as the level of difficulty some problem is in regards to the student's expected level of understanding. If it is more difficult, it has greater rigor.

The process of a course's peer-review is handled by smart contracts on this blockchain. Each course assignment (e.g. homework, project, quiz, etc.) is segmented into elemental parts, and each element's rigor score is determined by peer-review. Once a consensus is reached, the course is made available to students. This technology will expedite a course's peer-review, helping set an overall rigor for the course and its assignments. This also matches the course rigor of one institute with another, helping to establish an immutable record of academic credentials and what was specifically learned by each student.

This helps one to eliminate the task of measuring what the student explicitly learned to match them successfully with their next course at their new school or next job. By connecting these topics both generally (to a course) and specifically (to each part of an assignment), we can compare courses' contents easily by examining both of their clearly defined Knowledge Units and topics. Then, to further weigh the value of similar courses, one

can compare their rigor scores and determine if one course has a greater value and covers more material than the other.

Student assignments could also be evaluated on the blockchain via smart contracts. This would streamline the grading process and shorten the grading time, while reducing human error and bias. A customized algorithm is then used to calculate a proficiency score for each student for each KU topic, using student grades as well as the course rigor score within the parameters of the CAE framework. We will focus only on the CAE framework, and in the future, this should be expanded to include other educational frameworks.

Once a student completes a course, the proposed blockchain can be accessed by prospective employers who can select employees based on the scores in these KU topics. Using the skills required for a possible position opening, employers can specify the importance of a topic and match an applicant to an opening based on a compound score calculated from the student's proficiency in the relevant topics. This system enables the employers to efficiently check student records and verify the rigor level of each course and a student's success in a specific topic. Blockchain is an effective strategy to solve this problem in that each grade assigned can be modeled as a series of transactions, and it requires a consensus among both students and professors. The rules and regulations that are put into place in this system are outlined by the FERPA laws, and in turn, can be easily implemented in a smart contract. The auditability of the blockchain makes it easier for employers to review student transcripts, and in turn, make more educated decisions on hires, not only because of the immutability of the blockchain record, but also because of the rigor of the course. Results demonstrate that the system was successful in increasing the accuracy of hires through simulated data sets, and that it is efficient, as well as scalable.

For our research, we chose Ethereum [**72**]—the most popular distributed blockchain computing platform at the time (2019), which is equipped with smart contracts—the programs running on the blockchain. In process of our experiments, we realized that we needed a faster platform with lower requirements for the cost of blockchain computations. And hence, for our next implementation, we switched to the Proof-of-Stake based Algorand plat-

form [**16**], which is open-source, and which has a good community support. We believe that our micro-accreditation system can be easily ported into the Algorand platform—we leave it as a future work. We note that in our current implementation, we needed to trust the peer review process, specifically, we assumed that the course evaluators are honest. Since it might not be necessarily the case in practice, we needed a completely new automated reputation system. This led us to exploring decentralized reputation management in our next problem, since the reputation of the peers providing rigor scores should influence the calculation for an overall rigor for a course, its topics, and its tasks.

1.1.2. Problem 2: Reputation Management for Review Systems

In order to normalize the rigor of a course or an institution, the process requires coming to a consensus on a peer-reviewed rigor score for assignments and the overall course. The next problem to be addressed is that there is no ready-made consensus mechanism which ensures that a given peer-reviewed rigor score given is honest, unbiased, and trustworthy. As a sample application, we will choose reviewing products and services in decentralized marketplaces. Trust is one of the major concerns in any online marketplace framework. Major challenges in this setting include data privacy, trustworthiness, efficiency, and many other aspects which concern both buyers and sellers alike. In practice, all the parties with an interest in the marketing process may deviate from the prescribed procedures at any given point. Therefore, it is important to devise a mechanism that would allow us to gauge the parties' trustworthiness.

Customer feedback is a powerful tool, which is commonly used in trust management. Often, it is implemented as review systems. In many marketplaces, this feedback mechanism is recursive in the sense that it allows others to up- or down-vote reviews seen as helpful, that is to provide feedback on reviews. Some marketplace platforms assign a certain status to a party in order to enhance the credibility of their feedback. An example of this is Amazon's "verified buyer" status. Other platforms, such as eBay, use human-driven reputation systems to ensure some level of trust in either the seller or buyer [**59**]. This time-consuming process is typically prone to biases and errors, in part due to the massive quantity of data to be

evaluated. Note that it is still up to the buyer to evaluate products and sellers by manually going through the reviews and evaluating their accuracy—this adds an extra hurdle, now on the consumer's part. In practice, buyers usually rely on the top-rated (perceived as most helpful) or the most recent comments—which, in general, may not paint an objective picture. A natural question, which arises in this context, is what kind of mechanism may enable us to deliver an objective view of a product to the customer. In particular, we focus on the setting of decentralized marketplaces, which have been gaining popularity in recent years, such as BitBay, OpenSea, Ocean Market, Origin Protocol Markets, to mention a few.

It is worth noting that trust in the context of online marketplaces has a wide variety of aspects. For instance, the buyer trusts the seller to accurately present the product, to deliver it in a timely fashion, to properly process payments and reimbursements, and so on. These aspects have been addressed in a large body of literature, such as, e.g., [**13**, **57**].

In this work, we are focusing on a particular aspect of trust in reviews and reputation of the parties who provide them: review credibility. Specifically, we are focusing on the following particular scenario, which is the first step towards the above-mentioned mechanism: A party leaves a review (say, on a product or service), which consists of a text and a rating. We will use NLP to evaluate the "positivity" of the text, and then we will compare it to the rating. A trustworthy review is expected to have a good match of the positivity to the rating.

Next, we need a system that would provide an incentive for the reviewers' trustworthy behavior. A natural solution would be to apply a reputation system in order to leverage trust in the reviewers and their feedback. A moderator could be hired to handle incongruent ratings, as well as spam and fake reviews. They would be charged with processing reputation adjustments and dolling penalties for infractions. The challenges with this are the substantial human resources needed in both cost and time due to the manual handling, and being prone to bias analysis due to human handling. A better approach is to handle the review evaluation automatically using NLP, with the reputation and rating administration also driven automatically. Finally, in the decentralized scenario, it is natural to use blockchain

for storing the reputation values and updating them (automatically, using NLP evaluation) according to the reviewers' performance.

Our solution is an architecture for a reputation-based review evaluation system, which is built on top of the blockchain system, to ensure correct and trustworthy assessments. We focus on a particular aspect of trust in the reviews, and reputation of the parties who provide them—altogether, this ensures the review credibility. Additionally, we provide the incentive for the reviewers' trustworthy behavior. Since a peer review is like other types of review, we focused our problem on the lack of trust and review credibility in decentralized systems like a marketplace. In our proposal, the trustworthiness of reviews is evaluated using NLP and the reviewers are assigned a reputation according to this evaluation. The reputation is stored on the blockchain and is used as an asset for the consensus process. With what we concluded, we believe we could apply this solution eventually to help trust micro-accreditation peer-reviewed rigor scores in the future.

As the underlying blockchain system, we use Algorand, which we mentioned above. We use the "reputation as a stake" approach as in [**37**] [**61**], and map the reputation directly to stake. This allows us to directly use Algorand as the blockchain system. In testing, our simulation results showed that the NLP component incurred a reasonable delay this new type of review transactions. Additionally, we measured the time required to add a standard payment transaction in Algorand and that for our review transaction and observed comparable results. Also, we observed that the NLP component ensures an accurate credible evaluation (compared to the ground truth) of the product review texts. With this new model, we have moved towards showing how technology can be used to evaluate the trustworthiness of both the reviews and the corresponding reviewers. We deploy NLP to determine whether the reviews are congruent and trustworthy. Additionally, we show how NLP can used for self-regulating trust management in a decentralized marketplace ecosystem.

As mentioned above, our implementation relied on the underlying Proof-of-Stake blockchain system, Algorand, which is deployed at the blockchain layer. However, we discovered that our applications to review assessments require a certain level of flexibility and

fine-tuning, which the existing Proof-of-Stake consensus mechanisms do not provide. The main reason is that the existing PoS systems are mainly designed to support the digital payment systems. At the same time, the existing Proof-of-Reputation systems are either tailored for a specific application or too general to cater for the review assessment purposes. Therefore, a need for a new consensus mechanism arises, which will serve this particular scenario best.

1.1.3. Problem 3: A Fitting Consensus Mechanism.

In order to address the above mentioned problem, we present a novel consensus model called *Proof-of-Review* (PoRev), which borrows concepts from Proof-of-Stake (PoS) [**12**] [**15**] and Proof-of-Reputation (PoR) [**26**] [**6**]. The participants of the proposed blockchain system agree on evaluated reviews, which will be added as transactions to the system. The reviews and the related evaluation data are stored on the blockchain. The reviews drive the reputation model, where reputation is used to regulate a user's participation in the system. PoRev is a tool that can be used to aid applications by ensuring honest and unbiased reviews and assessments, hence providing an immutable and transparent record of data related to both the reviews and the reviewers.

The motivation to introduce such a system is to derive a trust in the participants' reputations through a consensus of their evaluated reviews. In doing that, applications could use this data to apply trustworthy weighted calculations towards an overall value of something or a rating. This system lends itself easily to decentralized marketplaces, where the trust in the reviews and reviewers is important to gauge the objective value of a product or service.

Our model could be used to mitigate some common challenges in a review system as bad reviews, spamming, bias, and human errors. The proposed model can be used in a wide variety of applications. For example, in micro-accreditation, the rigor of a course could be settled from the aggregation of peer-provided reputation-weighted scores. In sensor-array systems, a node's assessment being incongruent could indicate a failure in a sensor or its corruption by an attacker. The PoRev consensus can then be used to steer the influence

away from the failing/corrupt nodes and provide a "heads up" to an overall monitoring system. Furthermore, this concept can be applies to ad-hoc networks of autonomous vehicles, e.g., those that caravan together on a highway. Here, the PoRev consensus can be used for decisions on routing or speed based on the information on road conditions or behavior of neighboring vehicles, hence empowering a "well-behaving"—meaning well-configured—vehicles. The evaluation component is generalized to provide a conclusive binary answer to the questions, "does this proposed assessment look right, and do I agree the assessor is providing it in good faith, with honest intent and faithful performance?" This is evaluated as either a good review or a bad one, whether in the manner of congruence in a review-rating or other gauged conditions and validations. These are just a few examples of applications where this system can be applied, outside of the common review (as for products or services) systems, especially taking into account the community-based blockchain systems, such as neighborhood watch [**61**] gaining prominence.

We propose a Proof-of-Review system, which integrates the reputation system into Algorand's PoS engine, and we add the two useful components which enable blacklisting and enforcement of the minimum-reputation requirements.[1] Specifically, we update and modify the Algorand core to implement our new protocol.

A player may be blacklisted and prevented from participating in the protocol, as decided by a majority vote during the consensus. This can be applied to both reviewers or a selected block leader, depending on the type of malicious actions like spamming or deviating from the prescribed protocol. A certain minimum reputation is required to participate in the committee selection. It is chosen in a way that prevents an access by malicious (and hence under-performing in terms of reputation) users to participation in the system, while also preventing a condition where only a small percentage of high-reputation nodes skew the committee selection. An adversary, as with any new player, would need to put in some honest work to gain reputation in order to participate beyond submitting reviews.

[1]We note that the minimum stake handling in Algorand does not quite satisfy our goals. See Section 5.3.5 for discussion.

For security analysis, we argue that our modification to the Algorand core preserve the properties of the original PoS system up to adjustment of some parameters, so that the resulting blockchain system remains secure under an assumption that up to 1/3 of total reputation (instead of stake in the original Algorand) is controlled by the malicious parties. Specifically, we adapted the security analysis on Algorand to show that our modifications preserve liveliness and completeness of the proposed system.

Instead of using Algorand in the off-the-shelf model—as it is done in [**74**]—we modify the engine by generalizing the validators (we use NLP, but other evaluation systems can be used for other purposes), adding the Proof-of-Review mechanism and blacklisting. Since we assume that the participants belong to a community of users contributing towards a certain goal, the assumption that no more than 1/3 of the total reputation is controlled by the dishonest parties appears to be reasonable.

With our new consensus, we have also shown how our model prevents maliciousness and provides strong security guarantees for the data, participants, and reputation values. We employed blacklisting and the minimum-reputation requirement. We showed how our Proof of Review model addresses several types of attacks that are common for Proof of Stake and Proof of Review systems. We also presented a distributed application (dApp) to allow us to validate our system as well as to demonstrate the performance of blacklisting and minimum-reputation components.

In testing, our simulation results showed that our proposed blockchain system has liveliness and completeness properties. Specifically, blacklisting does not affect liveliness or completeness (that a transaction is guaranteed to be added to the blockchain eventually) when less than 75% of nodes are blacklisted. Additionally, blacklisting players for technical or behavioral maliciousness does not affect that round's time interval of when the honest players agree and know of that round's block. We also show that requiring a minimum stake to participate always shows liveliness and completeness (when properly tuned) regardless of the number of parties not meeting the stake requirement (as long as there are many enough left to satisfy the 2/3 honest majority).

1.2. Organization of This Thesis

This work is organized as follows. In Chapter 2, we provide the basic notations and definitions and also briefly describe the Algorand blockchain platform [**16**]. In Chapter 3, we discuss a solution for trust and assuredness in education systems to provide students the knowledge needed in the workforce. It is based on the work in [**76**]. In Chapter 4, we provide a system that will provide trust in the credibility of reviews (specifically in a decentralized marketplace). This result was initially published as [**74**]. Chapter 5 is devoted to presentation of the Proof-of-Review consensus. This result was presented in [**75**]. Finally, concluding remarks and open problems are discussed in Chapter 6.

CHAPTER 2

BACKGROUND

Let us briefly describe the blockchain systems which serve as the main building blocks of the proposed constructions.

2.1. Related Consensus Models

2.1.1. Proof-of-Work Blockchains

In a nutshell, the Proof of Work (PoW) consensus implements a "lottery" between the nodes (participants) of the system, where the "winner" gets to add the next block to the blockchain (a distributed and tamper-resistant ledger of transactions and states). Winning requires the node to perform some calculations to solve a computational puzzle, which is normally quite energy-demanding. This is also known as mining. The fastest miner gets rewarded to add a new block. This block is verified by the other miners who implicitly accept it by mining on top of it or ignore it. Once the block is added (and being mined on top of), it is supposed to remain there permanently, hence making its data immutable. The success of Bitcoin [**50**] made PoW consensus model popular. We omit the details of the blockchain operation and refer the reader to [**58**] [**42**] for details.

2.1.2. Proof-of-Stake Blockchains

Proof of Stake (PoS) is another protocol which is commonly considered as an alternative to Proof of Work (PoW). PoS has the same goal as PoW which is to pick a node that adds the next block to the blockchain. It uses a different mechanism to get to that goal by operating with a finite amount of coins (tokens purchased by a user) used as a stake in the overall system. Proof of Stake only considers a peer's stake of coins, which they possess in the system. Those nodes with the most coins in the system have the most stake. The higher the stake, the higher the probability of being selected to participate in the system (being a "miner"), and also having the most to lose financially if found to be acting improperly.

2.1.3. Proof-of-Reputation Blockchains

In Proof of Reputation (PoR), the protocol uses the users' reputation values in securing the network. It is similar to the PoS consensus as there is no mining process (puzzles to solve), but with the reputation of a node being used to determine block forging. Only those with the highest reputation act as participants, or trusted validators, in the system (creating blocks, signing blocks, etc.). In most PoR platforms, it acts like an advanced Proof of Authority network, where once a company's reputation is verified by external entities, it acts as a trusted authorized validator node to others externally verified. This also means that the nodes are not fully anonymous, since the reputation is derived from parties outside the network. Only those validated can participate. Any improper action risks that company's reputation, with possible financial consequences. We refer readers to [**73**] for those interested in further details.

2.1.4. Discussion

For discussion, our contribution —including Proof of Review (PoRev)— may be seen as a kind of special adaptation of Proof of Reputation, where the reputation system is only one component of the consensus protocol. Compared to PoR system of Gai et al. [**26**], PoRev reputation model and associated values persist beyond a round's block being added. We use the reputation values of the nodes at the beginning of a round, instead of the top-ranked node at the end of the round. Our nodes' reputation values are calculated from each node's history of submitted transactions since initially joining the network, instead of calculated only from the ratings given in each transaction within that round's block. Instead of transactions being ratings (reputation scores) given by other nodes' raters (humans), a review transaction in our system contains the review, the analysis of that review, and the direction on how to adjust the reputation of that reviewer (increase or decrease). Our review evaluation is done automatically with the purpose to exclude a human factor (e.g., a possible bias). Furthermore, since the reputation of the nodes is not validated by external parties, the identities remain as anonymous as they would on a common PoS platform. Additionally, in discussing PoS, one can see that PoS is not associated with reputation and hence it cannot be

directly applied in our scenarios. A natural similarity to these types of systems is seen when directly mapping reputation to stake (as it is done, e.g., in [**60**, **74**]). Then, the components such as committee selection and block mining work in a similar manner as in PoS systems. Some major differences from the currently popular PoS systems is their focus on financial applications rather than community-based applications (such neighborhood watch [**61**] and similar ones such as data cooperatives—see the references in [**61**]).

2.2. Notations and Definitions

Definition 2.1. (Block). A block is an immutable structure of data representing the state and a payset of transactions of a system for a specific round. When linking on a blockchain, a block B also contains a record of the previous block's cryptographic hash, which is used to create its own digital hash identifier, unless it is the Genesis Block B_0. The block B is added to the ledger at the end of each Round r. In the circumstance where an agreement can't be made within a set time, or other cases, an empty block is voted on to be added to the ledger. An empty block has an empty payset (set) of transactions.

Definition 2.2. (Blockchain). A blockchain is a linked series of blocks starting with the Genesis Block B_0, showing an immutable history of a system's states and transactions. Mostly used as a distributed ledger, in which every peer maintains the exact copy of the ledger. A blockchain platform is also a decentralized system of nodes, all which maintain the same ledger of transactions and contracts within those blocks. Compared to a centralized system in which information retrieval could be bottle-necked, or prevented (temporally or permanently), the same transaction data and ledger are available from each node. If one node disappears, the others can still be accessed. A blockchain has three extremely important properties: immutability, auditability, verifiability. It is computationally hard to modify a block after a set number of blocks, auditable in that it represents a transparently traceable sequence of changes to the network, and verifiable in that all blocks must be verified before being appended to the blockchain. This ensures the validity of the block once appended. For further details on blockchain, we refer the reader to [**36**].

Definition 2.3. (Leader). A leader ℓ^r is a stakeholder whose proposed block is selected for a round from all potential leaders. A leader can be cryptographically self-selected to be a potential leader and then again for any step in the protocol as a verifier. The leader's block is assembled from review transactions, in which these transactions been evaluated by them in step 1.

Definition 2.4. (Potential Leaders). Potential Leaders (PL) are cryptographically self-selected to assemble a block of transactions for a round. They evaluate the review in each transaction and update the review fields in the transaction header before adding it to the payset of the block. Once the block has reached its size or time-limit, they then broadcast the block and their credentials to the network to potentially be chosen as the leader. A participant can only be in PL if they meet the minimum stake and are not currently blacklisted for that round. Like in other PoS systems, the larger the stake a peer has, the more likely to be selected as a leader.

Definition 2.5. (Round). Rounds start after an initial block (Genesis Block), peers, and reputation have been determined. The round is a set of 5 steps (simplified for definition sake, its process has a repeating set of steps until a consensus is achieved or time runs out). At the end of each round, a block will be added to the ledger, regardless of transactions within the payset of the block. In certain cases, the block may be empty. In the best-expected case, the review transactions of a block are finalized at the end of the round with the reputation adjustments completed and the block added to to the ledger.

Definition 2.6. (Genesis Block). The Genesis Block is the initial block B_0 containing the list of initial online participants with their respective identifying information including their initial reputation of 1 and their stake, which is initially distributed equally.

Definition 2.7. (Stake). A stake is a proportional advantage in the system when participating in the consensus protocol. User's i stake is calculated from their current reputation $\mathcal{R}$ for round r ($\mathcal{R}_i^r$) and is balanced against the currently online peers' reputation. Specifically,

$$\omega_i^r = \mathcal{R}_i^r / \mathcal{R}_{PK^r}^r,$$

where $\mathcal{R}_i^r$ is reputation of a user i, PK^r represents the public keys of all active users at round r, and hence $\mathcal{R}_{PK^r}^r$ is the total sum of reputation of all active users at round r. Simply, a user's weight is calculated from round r reputation for that user i divided by the total reputation in the system of all active users.

Definition 2.8. (Participant, Stakeholder). A participant is an online peer with a reputation greater than zero. They are identified by their public key (pk) and are a stakeholder in the system. They may be selected to participate in the consensus process for a round in one or multiple steps. Their stake is dependent on their reputation proportional to the total reputation of all currently online (non-blacklisted) participants. Though a participant may never submit a review, with their reputation remaining stagnant at a value of 1, they are considered a potential reviewer. Since their stake is based on their reputation, it is probable that they are never selected to participate in the protocol but will not affect their other types of actions (e.g. payments) on this platform.

Definition 2.9. (Verifier, Set of Verifiers). A verifier is a part of the Set of Verifiers (SV) cryptographically self-selected for each round in each step. Selection is limited to participants with a minimum stake and not currently blacklisted. A new SV is selected for each step, and Verifiers are tasked with different actions at each step in the round. In step 1 they are referred to as both Potential Leaders and a member of $SV^{r,1}$ and propose blocks. In step 2, they select the leader. Steps 3-5 are discussed in more detail later. Verifiers, in addition to validating each transaction technically in the leader's block, will validate each review transaction through evaluation (in the same way the leader has). They then compare this against the leader's evaluation to ensure the correctness of the evaluation and thus the block. When a discrepancy is seen, the verifiers vote to blacklist the leader. If the majority concludes the same, then the leader is blacklisted.

2.3. Overview of Ethereum

In this section, we briefly discuss the Ethereum blockchain platform as we use it in our micro-accreditation system presented in Chapter 3. We refer interested readers to [**71**], [**32**], and [**70**] for more details. The Ethereum platform uses a Proof of Work verification method for blocks. Whenever a transaction is to be completed, it is sent with a specific amount of ether as well, in order to compensate the miner, or the node executing the transaction, appropriately for the amount of computational power expended for executing the transaction.

On this platform, the Ethereum Virtual Machine (EVM) is used as the distributed computational environment where smart contracts can be executed or deployed. Ethereum transaction rules are codified in these "smart contracts", which are like programs that run when specific conditions are met and helps ensure multiple parties complete their side of the contract (or meet the conditions required to complete the contract). Whenever a transaction is to be completed, it is sent with a specific amount of ether as well, in order to compensate the miner, or the node executing the transaction, appropriately for the amount of computational power expended for executing the transaction.

The amount of computational power used is determined by calculating the gas consumed by the transaction. For details, we refer the reader to the gas calculation algorithm outlined in the Ethereum yellow paper [**72**]. Because the amount of gas consumed is determined by the number of opcodes (individual instructions of what to do) as well as the amount of data being handled by a function, it is safe to say that the amount of gas consumed by a transaction is a measure if its efficiency.

An Ethereum transaction "smart contract" can be accessed through a web3.js API, through which users can send transactions. Recall that smart contracts are agreements between two entities that automatically do something when certain conditions are met. A fast food kiosk is a simple kind of a smart contract that can be explained in a way outside of blockchains. You press what you want to order, pay with credit card, and the system delivers you the food. The contract is programmed to ensure you get food only after you have finished certain instructions.

In Ethereum, a wallet is needed to hold the ether (a cryptocurrency), and this ether is used to send/transact transactions (also known as the transaction cost). An Ethereum wallet that contains ether can be accessed through applications like the Mist Browser and Metamask and is also available for the user to access. Truffle, a development framework for Ethereum distributed applications, also supplies a wallet. A digital wallet is used to hold the keys to the cryptocurrency instead of the actual coins. The public key is used as an address where a user can send coins to and receive from another. The private key is used as a kind of password to access and control the cryptocurrency. Ethereum provided three globally-accessible networks: Mainnet, and two testnets, Ropsten [**5**], and Rinkeby [**65**]. Truffle also provides a network emulator, known as Ganache [**4**], for testing purposes. Ganashe is typically used before deploying a dApp to the testnets, since it is a personal, simulated, self-contained blockchain network with no other outside users. This allows one to test functionality without the wait times before eventually moving on to the testnets for performance tests. The testnets are used by developers to test their applications without requiring or using actual ether, though you are sharing this network with other users (aka nodes) also testing their own dApps. These other nodes will compete and be incentived like those on the Mainnet, so one can test and tweak their code for better performance. Times and cost do not reflect the actual numbers compared to being on the Mainnet, but do give the developer a sense of the performance when eventually deployed there.

2.4. Algorand Overview

In this section, we briefly discuss the PoS-based Algorand blockchain system and its mechanics, and we refer the interested reader to [**16**] for details. Algorand is an open-source, scalable, and decentralized blockchain that is applied as a payment system.

Algorand's consensus is an asynchronous protocol which is organized in rounds. By the end of a round, a committee will come to a consensus on a block to be added to the ledger. Each user is identified by their public key and has an associated amount of digital currency (tokens) to make payments. A new user can join the network when an existing user pays a new user tokens, creating a new account.

In this section presentation, we closely follow the presentation of [**16**]. At the start of network, the initial state of the system is publicly known, with a sets of users' public keys and associated amount of money. Round r starts by randomly selecting and publicizing (the identity of) a user, ℓ^r, the round leader. The leader constructs, digitally signs, and propagates a block B, which is their own candidate for the r-th block and which includes a set of new and valid payments. Next, a small set of selected verifiers, SV^r, referred to as the committee, is randomly chosen and publicized. The size of the committee is such that, with the overwhelming probability it has at least a 2/3 honest majority. The committee reaches a Byzantine agreement on the block B proposed by ℓ^r. Upon termination, each honest verifier locally outputs his own block, whose hash value he digitally signs and propagates. Block B^r is defined to be the block that has been signed by a given number of properly chosen verifiers.

The following five steps present a simplified view of the Algorand protocol, highlighting the steps, which are important in the context of Proof of Review. There are other virtual steps that form a cycle from the last two steps to ensure convergence, but they are omitted for clarity since they are not directly relevant to our proposal. We refer the interested reader to Section 4.2 of [**16**] for details.

In the below description, when we say that a participant "cryptographically self-selects" themselves, we mean that they use a cryptographic self-selecting mechanism as described in Section 1.2 of [**15**] (cryptographic sortition) for constructing a committee of verifiers and potential block leaders at each step. This functionality is based on the results from a Verifiable Random Function acting like a weighted lottery, which uses a participant's key, stake, and the round's selection seed to return a string indicating committee membership. This membership is verifiable by others on the network. Additionally, Algorand restricts participation to nodes existing at round $r - k$, where k is the set number rounds before the current one. This prevents new account activity, regardless of the percent of tokens the account owns.

Step 1: Block Proposal. First, the participants cryptographically self-select themselves

to be Potential Leaders (PL) for the round. These PL assemble a block by verifying that the transactions are properly formed before adding them to the payset (a structure in the block that is the set of payment transactions) of their potential block. The block is signed, then PL propagates a message, including the signed block and their hashed credentials, to the network for a soft vote (next step).

Step 2: Block Leader Selection (soft vote). Again, participants cryptographically self-select themselves to be a verifier in the current round's set of selected verifiers (SV). Each player in Step 2's SV seeks to determine the round leader by listening to incoming messages and comparing the hashed credentials to all other hashed credentials received so far. After a certain amount of time, the leader is determined by the message with the least hashed credential value. Each player i in SV will then propagate their vote for this leader (and block) in a message.

Step 3: Block Verification (certify vote). As in previous steps, participants cryptographically self-select to be a verifier. Each player in Step 3's SV seeks to verify and validate the leader's block by iterating over associated transactions within the payset and performing technical checks, such as checking if the transactions are properly formed, validating the corresponding signatures, and so on. Once all transactions have been verified and validated (certified), the player propagates a cert vote message for that block [**7**]. If any transactions are considered incorrect, then the message is a vote for an empty block instead.

Step 4: BBA begins (GC to BBA). We simplify the explanation of steps 4 and 5 for clarity since the Graded Consensus (GC) and Binary Bynzatine Agreement (BBA) are not important or relevant elements to our specific research proposal, and are used technically to converge to agreement and end a round due to the asynchronous nature of their network. Again, we refer readers to Section 4.2 of [**16**] if interested. Each player in Step 4's SV wait a maximum amount of time to receive the minimum number of messages (from Step 3's participants) who are in agreement on the leader and respective block.

Their next message will be similar to the previous steps' messages, but it will include a flag (b) that indicates whether they have heard the minimum number of votes that are still in agreement for the leader and block (true), or not (false).

Step 5: Block Decision. Each player in Step 5's SV seeks to come to a final decision on a block by converging on an agreement on the leader and block.

If they receive the minimum number of messages from Step 4's SV with the flag (b) being true, then the round concludes. If the block is not empty, the transactions within the block will be executed before the block is added to the ledger. Otherwise, a cycle of virtual steps begins (referred to earlier and omitted for clarity) to eventually come to consensus.

We chose Algorand as the underlying blockchain system, because its consensus model was easy to adapt to add additional components for our purposes. Additionally, it is open-source, easy to modify its engine (the Proof-of-Stake mechanisms) such as the committee selection, stake calculation, and other mechanisms for our purposes, and it has a rich community support.

CHAPTER 3

MICRO-ACCREDITATION FOR MATCHING EMPLOYER E-HIRE NEEDS

3.1. Introduction

We explore the issue of trust in education systems to provide knowledge and skills needed for the workforce. How can we trust courses in one college to provide a student with the type and level of knowledge needed in the workplace? We present a decentralized micro-accreditation system to track student records throughout an academic process at schools and to help matching employers' requirements to students' knowledge. Specifically, the system will help establish immutable proof of a student's academic credentials, what one specifically learned and how well they learned it, by breaking down each course into topics, knowledge, and tasks, then storing it on a blockchain. The student's records are stored in an Ethereum-based blockchain network, along with a decentralized school system where all courses, assignments, and assessments are accessed and normalized across the schools participating. Each school's course has a required set of Knowledge Units (KU) and micro-accreditations (topics) to be fulfilled comprehensively via assignments. In our case, we use the KU topics defined by the Center of Academic Excellence (CAE) [**31**]. Each assignment element will be tagged with both a KU topic and a rigor score, with the rigor determined by peer-review. These student assignments could also be evaluated on the blockchain via smart contracts. This would streamline the process and shorten the grading time, while reducing human error and bias. A customized algorithm is then used to calculate a score for each student for each KU topic, using student grades as well as the course rigor within the parameters of the CAE framework. Rigor is defined as the level of difficulty some problem is in regards to the expected level of understanding of the problem-solver. If it is more difficult, it has greater rigor. In this work, we focus only on the CAE framework [**31**], but in the future, the proposed system can be expanded to include other industry frameworks. This result was initially published as a paper titled "Micro-Accreditation for Matching Employer E-Hire Needs" in the proceedings of 2019 2nd IEEE International Conference on Blockchain [**76**].

The process of a course's peer-review is handled by smart contracts on the Ethereum blockchain. Each course assignment (e.g., homework, project, quiz, etc.) is segmented into logical elemental parts, with each elements' rigor score determined by peer review. Using this technology will help expedite a course's peer-review, helping to set an overall rigor for the course and its assignments. This also matches the course rigor of one institute with another, helping to establish immutable proof of academic credentials, what was specifically learned by each student, and how well it was learned.

This school blockchain can then be accessed by prospective employers who can select employees based from the scores in these KU topics. Students can also get certified based from the scores calculated for each topic. Using this, it becomes much easier for employers to comb through student records and verify not only the authenticity of courses, but the rigor of each course and a student's success in a specific topic. Blockchain is an effective strategy to solve this problem in that each grade assigned can be modeled as a series of transactions. The rules and regulations that are put into place in this system are outlined by the FERPA laws, and in turn, can be easily implemented in a smart contract. The auditability of the blockchain makes it easier for employers to review student transcripts, and in turn, make more educated decisions on hires, not only because of the immutability of the blockchain record, but also because of the rigor of the course. Results demonstrate that the system was partially successful by demonstrating it has potential to increase the accuracy of hires through simulated data sets, and that it is efficient, as well as scalable.

3.2. Problem Description and Requirements

Let us first identify several problems with existing hiring methods. We note that similar problems arise when students transfer to different schools.

The current hiring process is resource-heavy and somewhat cumbersome, as it involves a lot components, such as recruitment processes, interviews (sometimes a series of them, which go in rounds), and paper cuts. Much of this overall process is often outsourced to expensive trusted third parties such as recruiter companies, and it always involves the human resources staff. Much of the complexity and confusion in finding the right candidate-

match arises from these hiring parties being too removed from the actual working duties and required skills. Hence, these parties do not fully understand the needs of the hiring manager. Currently, the hiring process lacks regular, organized, and agreed-upon definitions of knowledge and skills, encompassing both academic performance and the workforce expectations.

It is difficult to locate information on new graduates in order to deduce their success in the workplace. Most info about them is contained in transcripts, which are difficult to interpret correctly from only course grades and the school name. This information is not normalized from one institution to another, and rarely reflects what specific knowledge was learned. Also, comparing courses between schools is very challenging. Accuracy is also an issue, since the skills required for a career is not matched to the courses accurately, as different skills are taught in different courses at different schools.

It is also difficult to discern if any bias existed during grading. A student's grade for an assignment or overall course could have been influenced by bias (i.e., how good was the relationship between teacher and student), moods of the grader, general oversight or carelessness.

It is worth noting that making mistakes in hiring is both risky and expensive, sometimes costing up to $240K due to recruitment, replacement, lost customers and brand image, re-ramping up for projects left behind, etc. as listed in [**23**, **24**].

Due to the above-listed reasons, it is of tantamount importance to create a system that efficiently and accurately reduces the risk involved in the hiring process through an effective matching algorithm, normalizes grades as well as the rigor of an institution, and eliminates middlemen involved in the process. Since the rigor is peer-reviewed, it reflects the most accurate standard in the score determination. In addition, combining rigor and KU topics creates a better solution for comparing courses from different academic institutions for school transfers. This helps eliminate the task of measuring what the student explicitly learned to match them successfully with their next course at their new school or with the skill-set required at the next job.

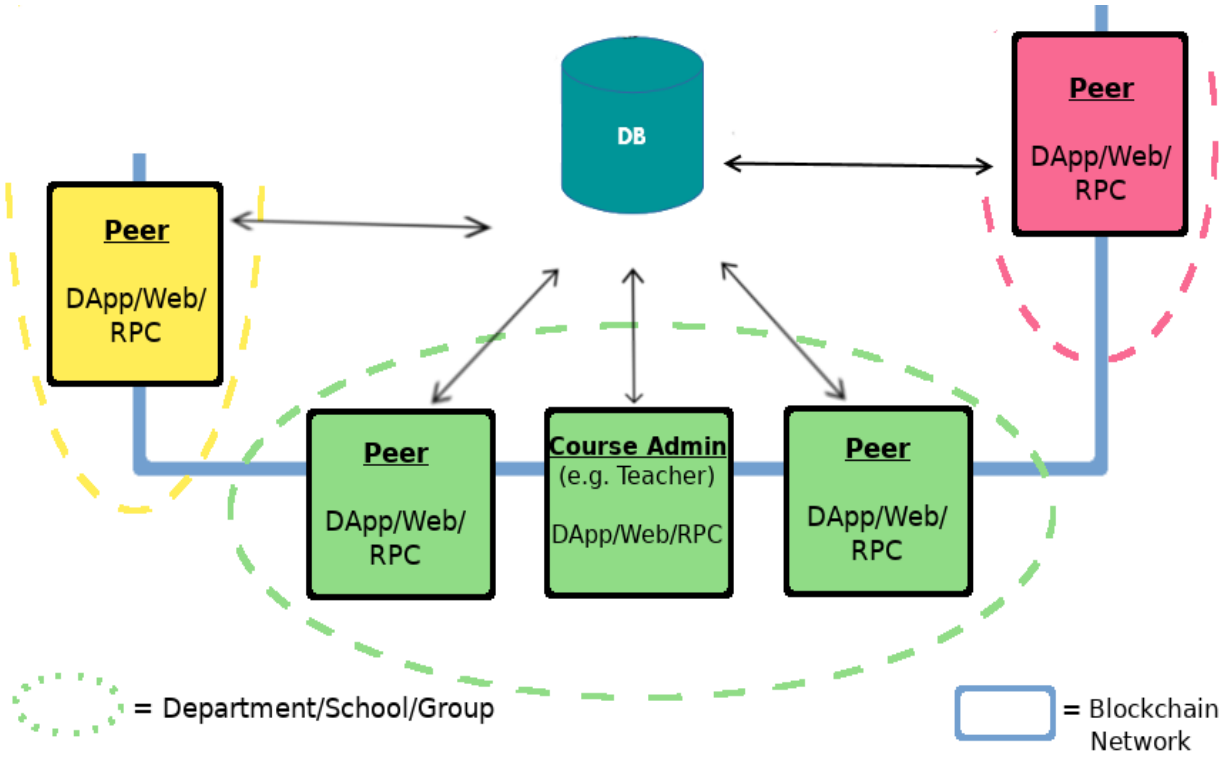

FIGURE 3.1. Possible Peer Review using Peer Groups.

By connecting these topics both generally (to a course) and specifically (to each part of an assignment), we can compare courses' contents easily by examining both of their clearly defined Knowledge Units and topics. Then, to further weigh the value of similar courses, one can compare their rigor scores and determine if one course has a greater value and covers more material than the other.

3.3. Related Works

Learning Machine's [**67**] applications use proof of existence in order to ensure the validity of credentials. Using Blockcerts, Learning Machine's technologies are currently used by the Massachusetts Institute of Technology in order to verify degrees and transcripts. However, this is different from our application, as it only serves to verify the authenticity of a record, and not effect on hiring specifically.

Chronobank [**18**] operates similarly to the above, in that it streamlines background checks, and it is more structured towards hiring and loading the transcript into the blockchain. However, this application is not oriented towards predicting success in a specific career and

matching students to a specific job.

TeachMePlease [**69**] is more oriented towards micro-accreditation through customizable and learning units. However, a transcript is not stored in this case, as it is more of a marketplace for educational material.

The Holburton School [**63**] is a learning platform that uses blockchain technologies in order to create an auditable and verifiable transcript in the form of transactions. However, this does not help with hiring, and is structured like any other traditional course, so that micro-accreditation is not covered.

Woolf University [**44**] is another learning platform that follows a scheme similar to the above, in that it is auditable and verifiable transcript-wise. However, there is no micro-accreditation due to a lack of peer review, and it does not help students or employers in the search for careers.

In the work [**53**], Ocheja et al. propose to place all student record data on the blockchain and to normalize it, regardless of origin. However, their "Secure Box" transforms the data from each Learning Management System to uniform data records for each student, which is seemingly very resource heavy, so the cost may not be scalable.

Hu et al. [**30**] present the idea of a "custodian contract" that spurs and manages new contracts. This is similar to the role of a course in our proposal. Using the course as a custodian that generates new smart contracts between students and assignments is something that naturally aligns with our proposal.

The work by Diana et al. [**19**] is related to auto-grading systems, specifically for their programming. However, their solutions do not use blockchain. This could be adapted into our proposal in the future and integrated into smart contracts for auto-grading.

3.4. Proposed Architecture

The proposed system is based on a traditional web-based dApp architecture. The frontend is web3.js (a Java library), using RPC to interact with the smart contracts on the network.

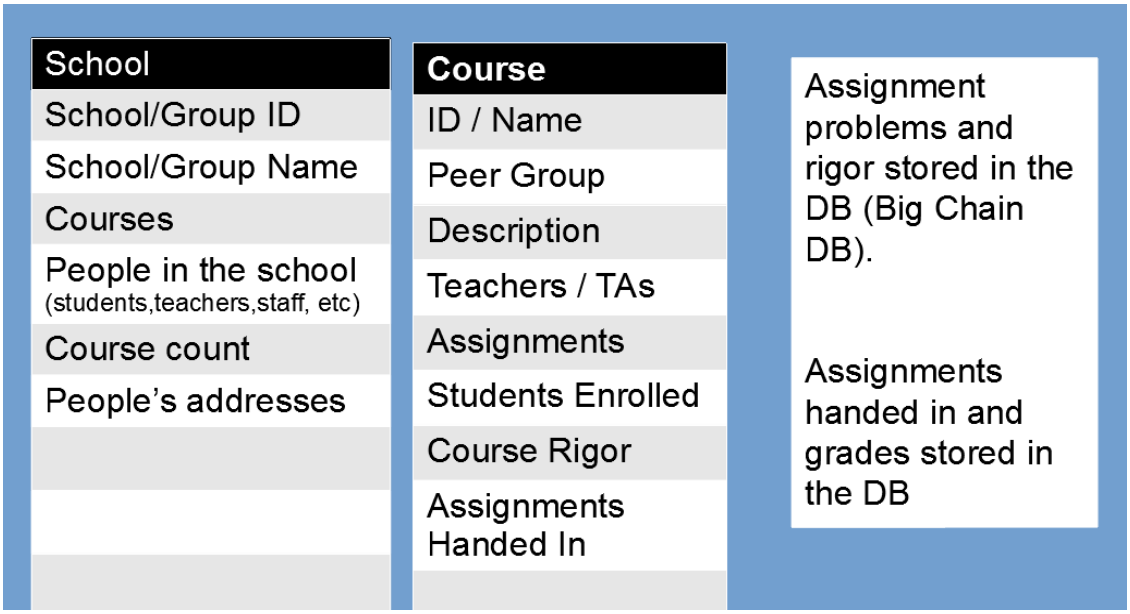

FIGURE 3.2. Structure of the Ledger for a School.
Note: What is stored on the blockchain in the ledger. In the future, for better efficiency, most of this will be stored off-chain possibly in BigChainDB or other system.

Essentially, javascript runs the frontend Ethereum Universe which is coded in HTML, connecting this web3.js-infused webpage to the Ethereum blockchain in the backend (Fig. 3.5). We chose this because it was the most common and most supported way to deploy a dApp outside a mobile Android app. Web3.js is also utilized to interface with IPFS, a decentralized filesystem in order to store assignment files, as storing files on the blockchain directly is far too expensive. One difference between the proposed design and the final design of the dApp is that BigChainDB, a blockchain-based decentralized database, was not utilized due to lack of time. In the future, for efficiency and scalability, the structure of the ledger should be stored off-chain in that or another decentralized database.

3.4.1. Requirements Considered

Blockchain is able to address many of the aforementioned problems, as its auditability, immutability, and verifiability serves to eliminate the need for recruiters, auditors, and other hiring staff, while the consensus-based nature of a blockchain enables us to normalize courses, and the system itself would make the hiring process cheaper due to the lack of third parties,

as well as the reduced risk due to the streamlined matching algorithm.

A solution for the e-Hiring match component must eliminate the need for a third party and normalize courses. The workload required by a third party is picked up by all the nodes on the network. All blocks can be assumed to be valid, and all the transcript materials included are valid as well. It automates much of the work that HR does in terms of shortlisting candidates by grades. The system could normalize courses through the properties of peer review and consensus, as it enables all nodes to come to an agreement as to the rigor of the course itself. This will be factored into our grade calculations and skill calculations. The candidate's ability to succeed in a career is determined by the individual skills taught in the course, also determined by the peer review process.

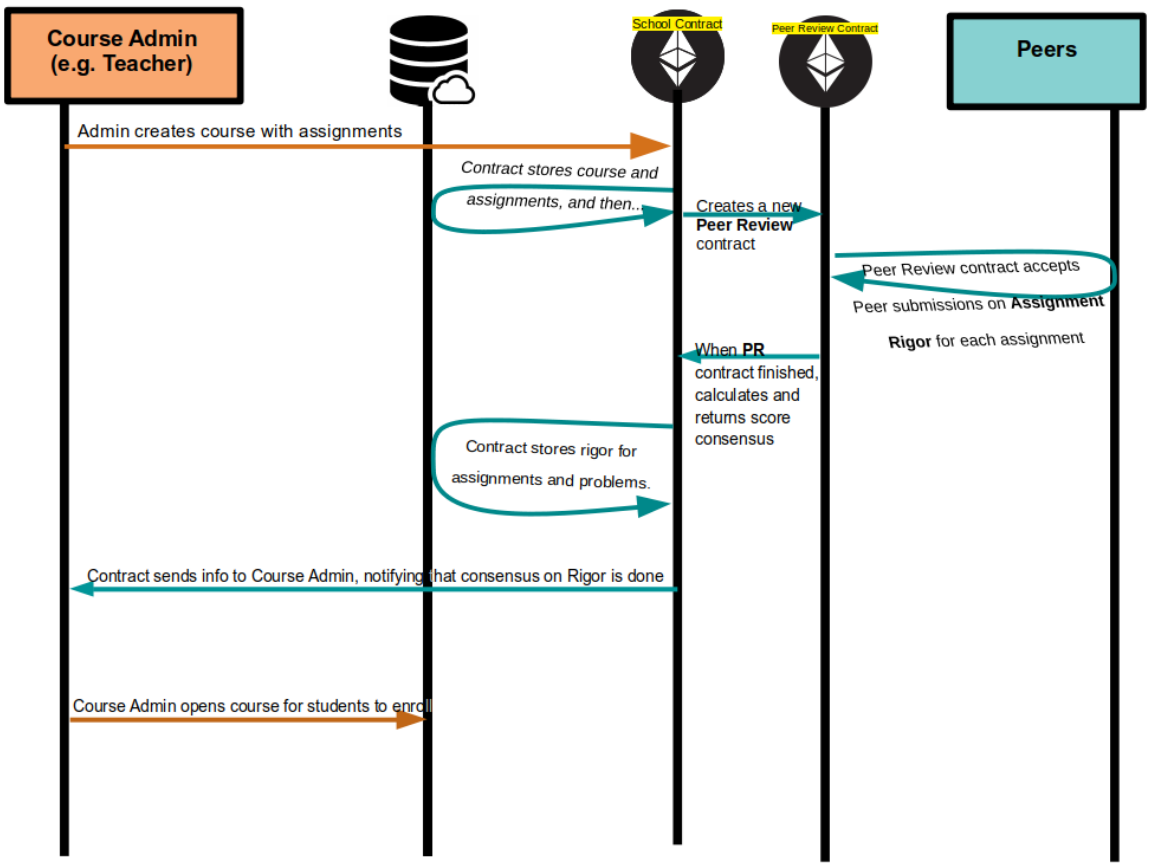

FIGURE 3.3. Work Sequence for a Peer Review.
Note: This sequence was used as the base for peer review work.

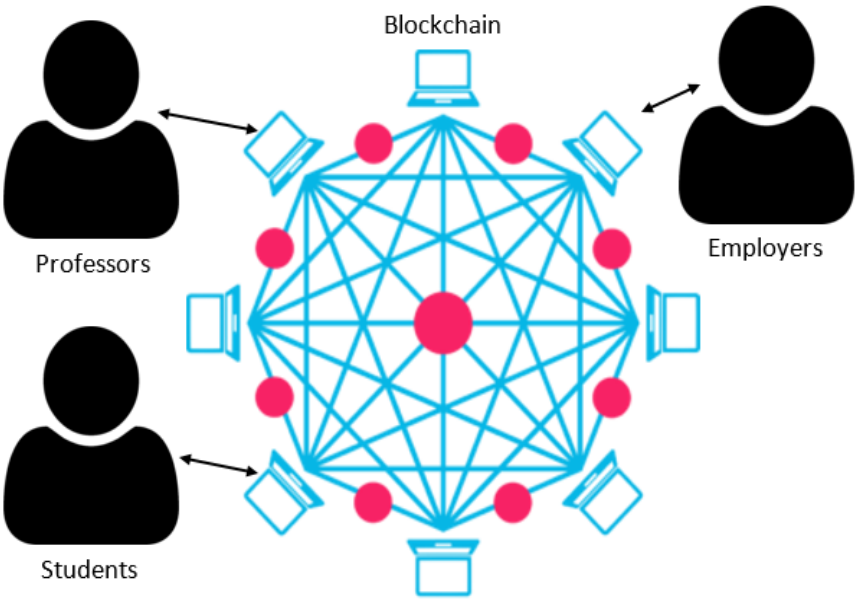

FIGURE 3.4. Micro-Accreditation Participants on the Blockchain.

3.4.2. Course creation and a peer-review component

Each course has multiple assignments. with multiple parts, and each part includes connected Knowledge Units, topics and a rigor score associated with it. The rigor of each problem (assignment part) will be peer-reviewed and determined through consensus. This consensus will most likely use a new consensus mechanism involving peer groups (Fig. 3.1). that will be explored extensively in future work. Currently we are doing this manually. These peer groups could be defined as follows:

- Peers with similar education and knowledge.
- Peers in similar field of study.
- Peers with PhD.
- Peers in same school or department.
- Peers teaching similar courses.

Once complete, the course is made available for student enrollment as shown (Fig. 3.3). Each participant will have their own node on the blockchain. The teacher, admin, and peers will have different web interfaces for interacting with the smart contract, and different access to the data stored.

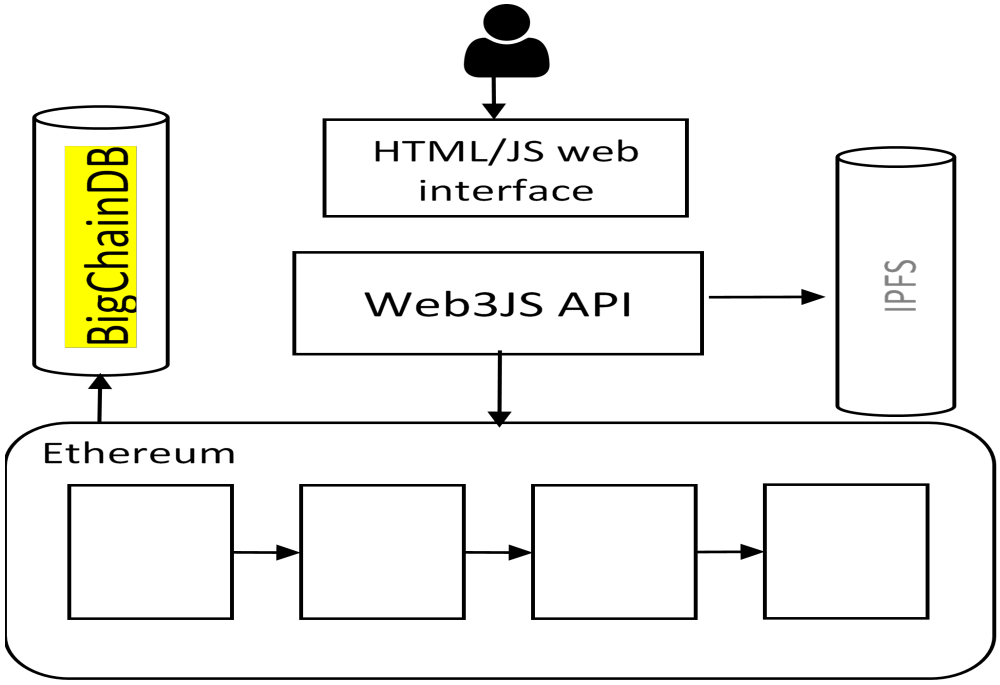

FIGURE 3.5. Architecture of the Proposed System.
Note: Web-based dApp using web3.js to connect to the contract on the Ethereum network. The contract connects to the BigChainDB for adding and retrieving data that is stored off-chain.

3.4.3. Detailed Design

The system participants are professors, students, and employers (Figure 3.4). The purpose of the inclusion of the students in the system is for them to complete coursework so that professors can assign grades to them. Professors are tasked with assigning grades and peer-reviewing courses, while employers are tasked with posting job opportunities. Once students graduate, they can apply for career opportunities, at which point all job applications are ranked by a compounded skill score calculated by the following equation (Equation 3.1), where s_k is the score for keyword k, $g_{a,k}$ is grade for assignment with keyword k, r_a is rigor for assignment a, and n_k is the number of assignments with keyword k:

$$s_k = \frac{\sum_{a=1}^{n} g_{a,k} - g_{a,avg} + r_a}{n_k}. \tag{3.1}$$

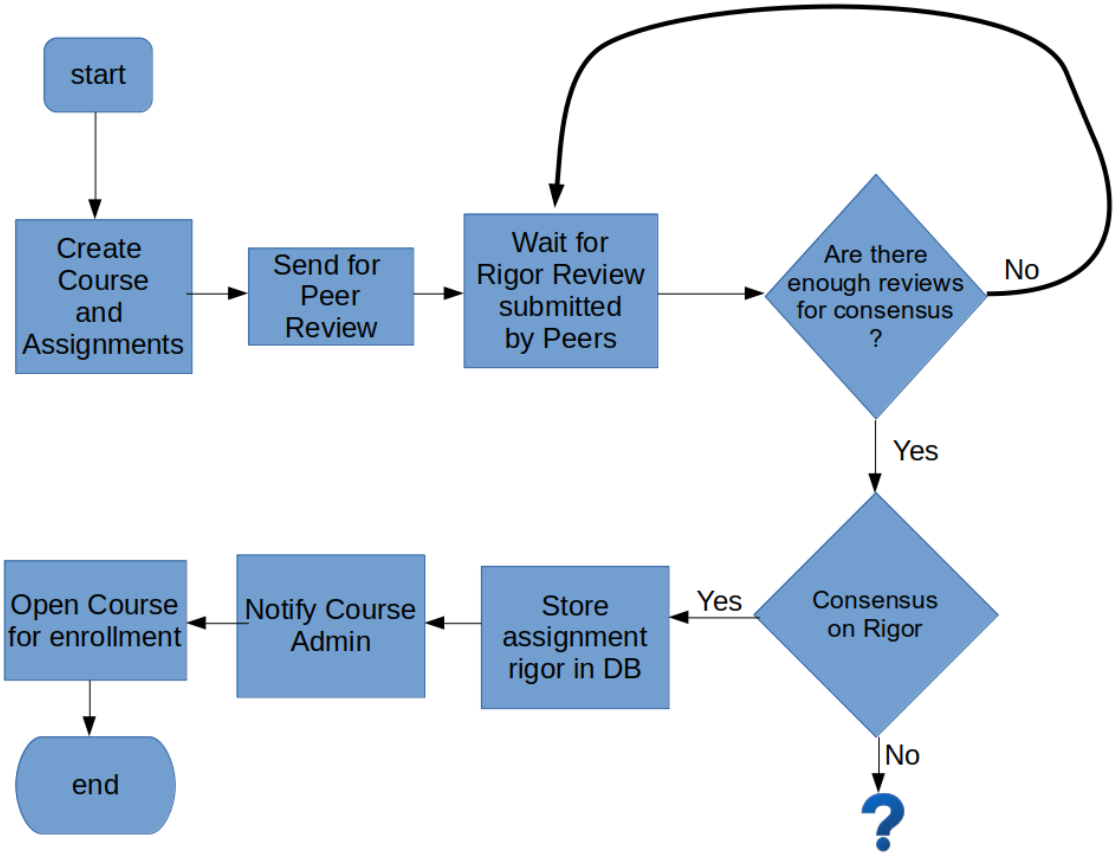

FIGURE 3.6. Work Flow for Peer Review.

For a job with k_{total} keywords, each with desirability d_k and skill score s_k the match index m is calculated as follows:

$$m = \sum_{k=1}^{k_{total}} d_k s_k. \tag{3.2}$$

A work flow was created for course creation and review showing where it starts (Fig 3.6). Note: In future work, we may need a third party contract to handle times it does not come to consensus for Peer Reviewed rigor, as well as a new consensus model for rigor.

3.4.4. Application for Course Peer-Review

This application methodology will proceed in the following manner. We are going to assume course is ready for review, that it exists to enroll in and is currently containing

assignments. The course admin (typically, the teacher) starts the process in the work flow, and the smart contract ends it once peer review consensus on rigor has been achieved. Once a course is complete (i.e., all assignments and parts are set and ready), the admin calls a transaction to "Complete Course".

Then, the admin calls a transaction for Peer Review. This generates a new Peer-Review contract between the admin and the peers. The admin furnishes the completed course, and the peers will provide their analysis of rigor for each assignment. Once a certain condition has been met (for example, a specific number of reviews have been entered), a consensus is made on the rigor. Once the peer review is complete, the course is open for either admin review or for student enrollment. The transaction with the course data is sent to the blockchain, and the smart contract executes it, while any assignment files required for the course are submitted to IPFS (see Section 3.4), and the respective references are stored on the blockchain.

3.4.5. Application for Enrolled Students

This application methodology will proceed in the following manner for all students. A student enrolls in the course, and their address is stored. As a student completes coursework, the grades are added to a mapping of their address to their grades. A student applies for credit, and a professor approves it. Upon approval, the student's associated knowledge scores are calculated and are stored in a global contract, accessible by only certified parties. A new course block is appended to the student's transcript blockchain.

A student applies for graduation and is approved by at least 3 professors. Once approved, a student then applies for jobs, and the match score is calculated by the algorithm mentioned above.

3.5. Experimental Methodology and Steps to Goal

The system was tested for operational success as well as efficiency.

3.5.1. Tests for Simulating the Hiring Data

We tested utilizing data sets and simulating hiring patterns. The data sets were publicly accessible grade distributions from three institutions (UNT, UT Austin, and UC Berkeley), as well as hiring statistics for one specific company (Amazon) found through LinkedIn statistics. The rigor for this test was determined using csrankings.org's CS rankings, and it was defined as $s = 50 + 50((100 - r)/100)$, where s is the given rigor rating, and r is the ranking given by csrankings.org. The success of this test was determined by the variance from the real statistics for last year. The independent variable here is the university selected, while the dependent variable is the selection rate of people from the university who are hired by Amazon.

The efficiency is determined by the amount of gas consumed by the algorithm. The amount of gas consumed by a function is proportional to its efficiency and the amount of computational work required by a node. Because of this, we measure the gas consumed by each function on the Ropsten network, in order to provide an appropriate testbed for the application, simulating the network load at the time of deployment. The independent variable here is the function type, and the dependent variable is the amount of gas consumed

The test for scalability was conducted by measuring the increase in gas price as the number of participants increases. This shows the decrease of efficiency of the program as the number of users increases and will also demonstrate exactly how the system responds to higher loads.

3.5.2. Tests for Simulating a Course and Course Peer Review

For the blockchain network backend, We used the programming language Solidity. Solidity is an object-oriented language used to write smart contracts. It's similar to other high-level languages, such as Java or C++, but does have many limitations since it was relatively new in 2019. This was used with the Truffle framework to allow easy compiling, linking, deploying, and migrating the contracts on an Ethereum network. We paired Truffle with Genache, development Ethereum blockchain that runs on your system providing you free ether crypto-wallets associated with each account (node) it sets up for you.

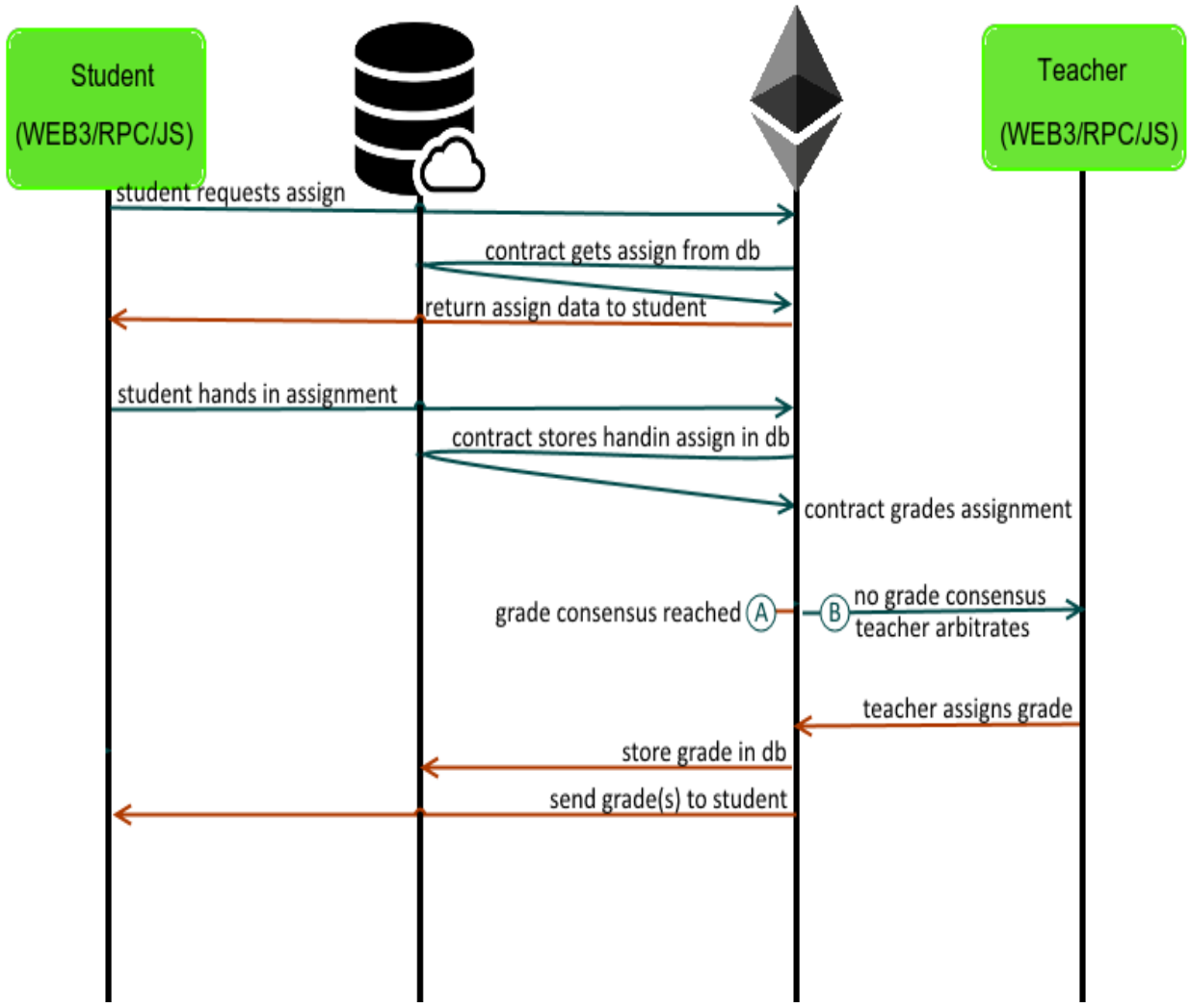

FIGURE 3.7. Work Sequence for Peer Review – The work sequence used in our simulations for peer reviewing. In future work, we may need a third party contract to handle times it doesn't come to consensus, as well as a new consensus model for rigor.

You need ether to transact on the network, as well as "gas", which Genache also manages.

In our setup, we manually created two courses with 3-4 assignments and 5 parts to each assignment. These were coded into files in the Solidity language, then JSON files were produced at build time from them. This was our test data. We used this test data to load these courses and assignments into the contract when starting the dApp, after an admin clicks the "Load Data" button. This established a base data set to start with. At

that point, since the assignments were now available, the Admin/Teacher could perform actions like closing course, calling for peer review, viewing the course. Additionally, for simplicity in grading, all assignments are exact answers and had no rigor assigned. We used a browser plugin named Metamask to help simulate different peers. MetaMask helps bridge the connection between the Ethereum network and the DApp your looking at by "seeing" the accounts created in Genache and will manage the "handshake" agreement for every transaction against the contract on the network. It also can be used on the Mainnet and Testnet if configured manually with public addresses given.

To simulate the peers, we set up a few peers and a course admin. After the test data loaded, we first ran through the dApp as the admin, closing, calling for peer review, recording the performance metrics for transaction time and gas cost. We compared multiple peer reviews both on a stand-alone simulated network, and the Ropsten Test Network for transaction times. JavaScript's performance.now() was needed to obtain transaction times for each transaction. A dependent variable might be Gas Price, which we discuss in the Results section below.

To simulate the students, we used the same test data for assignments and courses to simulate them enrolling and completing assignments. Using Metamask (similar to the Peer testing above), we created four students to run through the student interface to complete assignments in a course (Fig. 3.7). When completing a course, the student hands in the answers, with the contract processing them, bundling together and storing a copy of the question parts (which include the directions, KU topics, and the rigor score) with the student answers. This allows changes to the course's assignments without affecting the existing student completed datasets.

3.6. Results and Discussion

3.6.1. Metrics Used

The metrics utilized in our analysis of hiring include those reserved for efficiency, scalability, and operational success. The metrics utilized in our analysis of peer-review include gas cost, transaction times, showing the effect of gas price (incentive for the miners

to do the work) on transaction time. These metrics and their testing methodologies are outlined above.

To measure *success*, we simulate a representative dataset gathered from Amazon's LinkedIn hiring data utilizing data sets from UC Berkeley, UT Austin, and UNT, as well as rigor rankings from CSRankings.org.

To measure *efficiency*, we measure the amount of gas consumed by each transaction run on the network, as gas is a measure of the efficiency of a function.

To measure *scalability*, we measure the amount of gas consumed as the number of entities accessing the network grows, in this case, students. We track this using the Ganache emulator and use this to measure how different transactions respond to a high network load.

3.6.2. Datasets and Results

We compared the simulated and real hiring of Amazon based on data sets. The graph (Fig 3.8) shows that the current algorithm is somewhat capable of distinguishing hiring patterns, and that the new graduates' hiring rates from UT Austin and UC Berkeley are similar to that of the given data set. However, it is observed that the data for UNT in the simulated run is completely zeroed out. The error for this measurement can be attributed to the algorithms extra weight on the rigor of an academic program. Because of this, one can say that the current data set is not capable of properly representing the state of real-world hiring, and because of this, is insufficient to properly deduce the success of the hiring algorithm.

We focused on the efficiency of running transactions on the network next. The graph (Fig. 3.9) shows the growth of transaction gas cost as the number of entities on the network increases. The only transaction whose gas cost grows is the Approve-Credit transaction. This is due to the de-allocation of memory containing the student data structure. Once again, this problem can be fixed through the inclusion of BigChainDB, as most memory-management functions can be handled by BigChainDB. Because of this, we can say that the system is scalable to an extent as well.

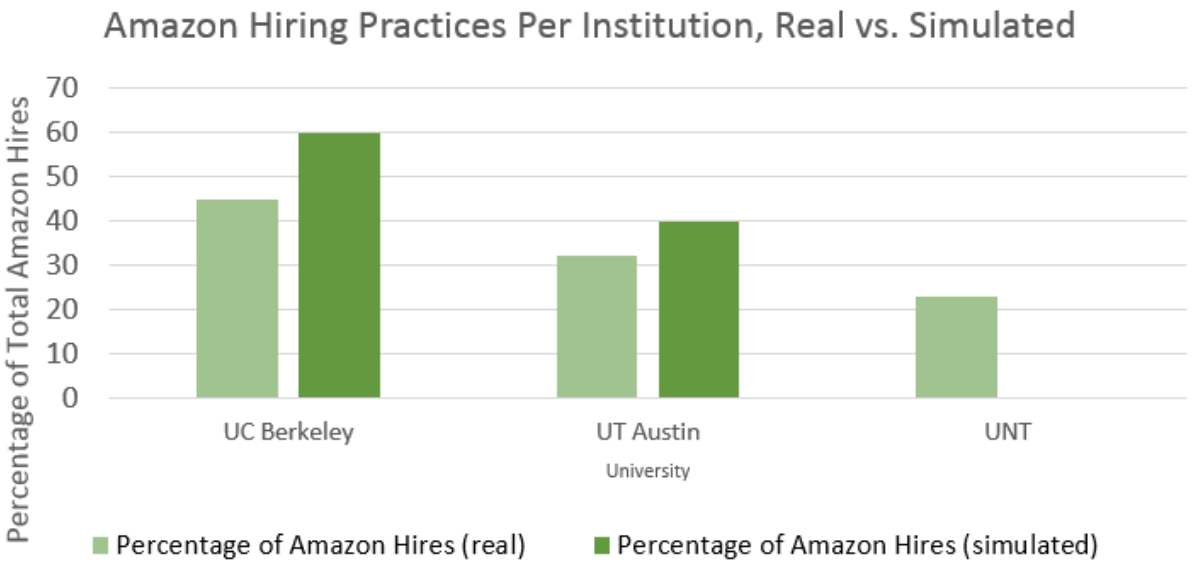

FIGURE 3.8. Hiring Practices.
Note: We utilized data sets and simulating hiring patterns. This graph shows that the current algorithm is generally capable of distinguishing hiring patterns

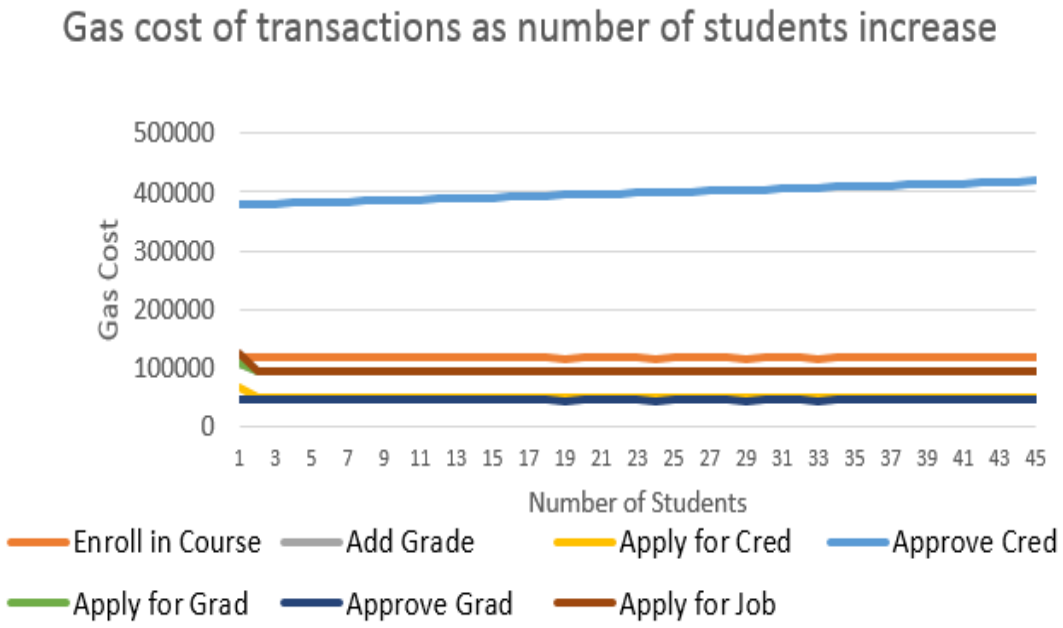

FIGURE 3.9. Gas Cost of Transactions.
Note: The transaction gas cost generally remains the same as the number of entities on the network increases. The only transaction whose gas cost grows is the Approve-Credit transaction.

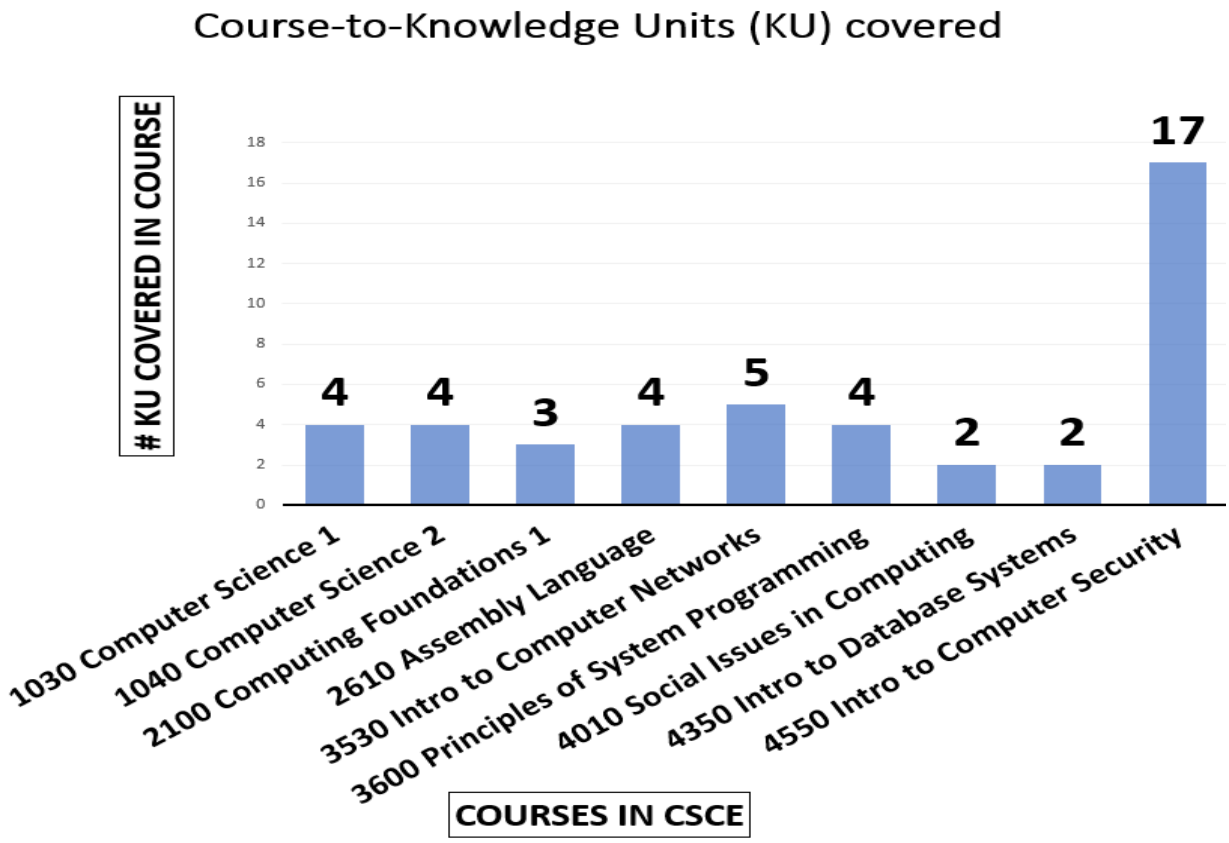

FIGURE 3.10. Knowledge Units mapped to UNT Courses
Note: A listing of some Computer Science Computer Engineering (CSCE) courses at UNT with their number of respective Knowledge Units (KU) [16] covered per each course. Majority of courses cover 3-4 KUs, with Intro to Computer Security covering 17. Each KU is broken down into topics (see Fig. 3.11).

Next we look at the number of Knowledge Units (KU) covered in some of the University of North Texas courses (KU defined by CAE framework). The graph (Fig. 3.10) shows the number of Knowledge Units (KU) covered – defined by the CAE Framework [**31**] – by each course in the Computer Science Computer Engineering Department. Only some courses are shown. These KU categories occasionally change and evolve (e.g. a KU that existed in 2016 might be split into multiple KU in following years). A KU consists of the base required topics to cover in addition to other information. The course Intro to Computer Security covers the greatest number of KUs in our chart, suggesting the course's breadth in knowledge covered.

We then look at the number of topics each Knowledge Unit covers. The graph (Fig 3.11) shows a general breakdown of the number of topics per Knowledge Unit. Some KUs, like IT Components and System Administration cover a lot more topics than others, as expected. To fulfill the requirements of the KU, one must typically satisfy all the topics and subtopics for that KU, defined by the CAE framework

We also explored the gas cost to deploy School and PeerReview contracts. The graph (Fig. 3.12) shows the gas cost is steep for deploying the School contract for peer-review. The School contract manages all the courses which all have assignments, as well as managing and interacting with the PeerReview contracts when necessary. The School contract has every function that can be transacted on by the user through RPC call from the dApp. There are additional functions (not shown in the graph) used in each library that are only called from another function.

Peer Review transaction times were then studied, comparing times when Gas Price (miner's incentive to do the work) is increased. The graph (Fig 3.13) shows the transaction times for the peer-review transaction. By default, when you call a transaction, the gas price is 1. The gas price (GP) is the incentive for a miner to work, and the higher the price is, the faster the transaction will happen. Changing the GP from 1 to 50 significantly and positively affected the transaction times. The Gas Price increase incentivizes the transaction. Manually adjusting the GP using Metamask is laborious due to there being many transactions and needing to change it for every transaction as they transact [**49**]. So we changed the transaction call in the code to reflect a specific GP of 50. We refer the reader to Section 3.5.2 for how Metamask was used and configured. In the future, testing should be done on the optimal price when needing a more immediate response calling the transaction.

We looked at the Gas Cost of each function in the Assignment Grading part. The zero gas costs are calls, meaning they do not alter any state (i.e., contract variables). The graph (Fig. 3.14) shows the gas cost for calling the Assignment Grading functions. The handinAssignment function consumes the most gas and will need to be worked differently in the future code-changes to significantly lower that cost.

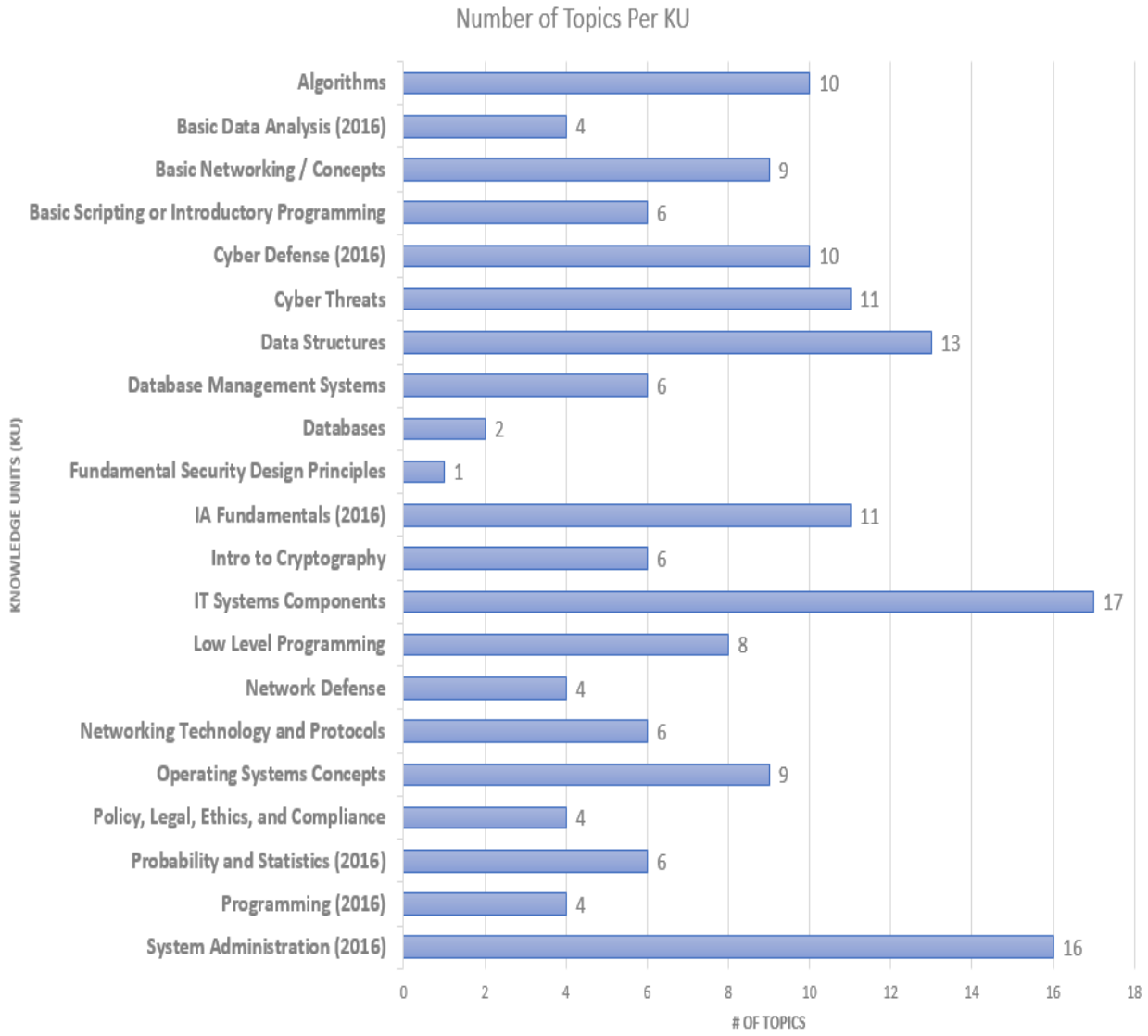

FIGURE 3.11. Topics Mapped to Knowledge Units (KU).
Note: Listing of KUs covered in list of CSCE courses (Fig. 3.10), broken down into the number of Topics covered in each. Both the KU and Topics are associated with each course and directs what is to be learned.

The current cost is associated with main transaction of handing in the assignment as well as all the "children" transactions associated with each part of an assignment (tasks). Perhaps an off-chain solution for all the parts of the assignment could be employed.

Finally, we analyzed the transaction times for the Assignment Grading. The graph (Fig. 3.15) shows the transaction times for the Assignment Grading. Similar to the Peer Review times, increasing the GP to 50, significantly lowered the times.

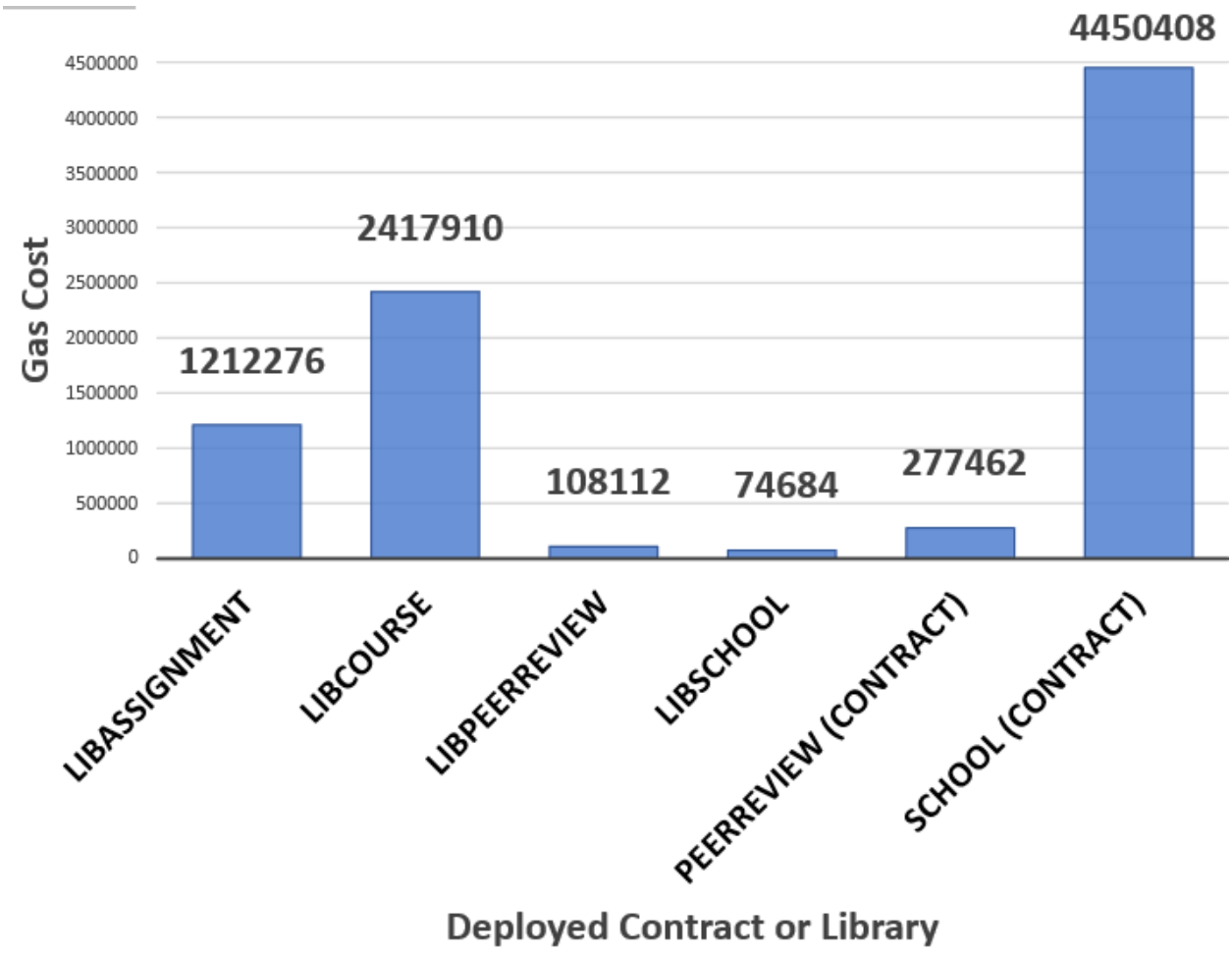

FIGURE 3.12. Gas Cost to Deploy School Contract.
Note: The initial gas cost on the Ethereum network to deploy the School and Peer contracts and the associated Libraries (also seen as contracts in Ethereum). After the initial deployment, any further gas cost is related to transactions (i.e. functions).

In example, at $GP = 1$, the "addAssignment" function times started at about 18 minutes to a much more reasonable 1.5 minutes (approximately). More research is needed to find a good balance for speed and gas price.

3.6.3. Discussion

Once a course and its assignments are completed, peer-reviewed with rigor calculated, and opened to Students, outside entities will be able to compare courses between different institutions (academic or other). The fuller picture would be established to gain better insight into a course, the course parts, what is to be learned.

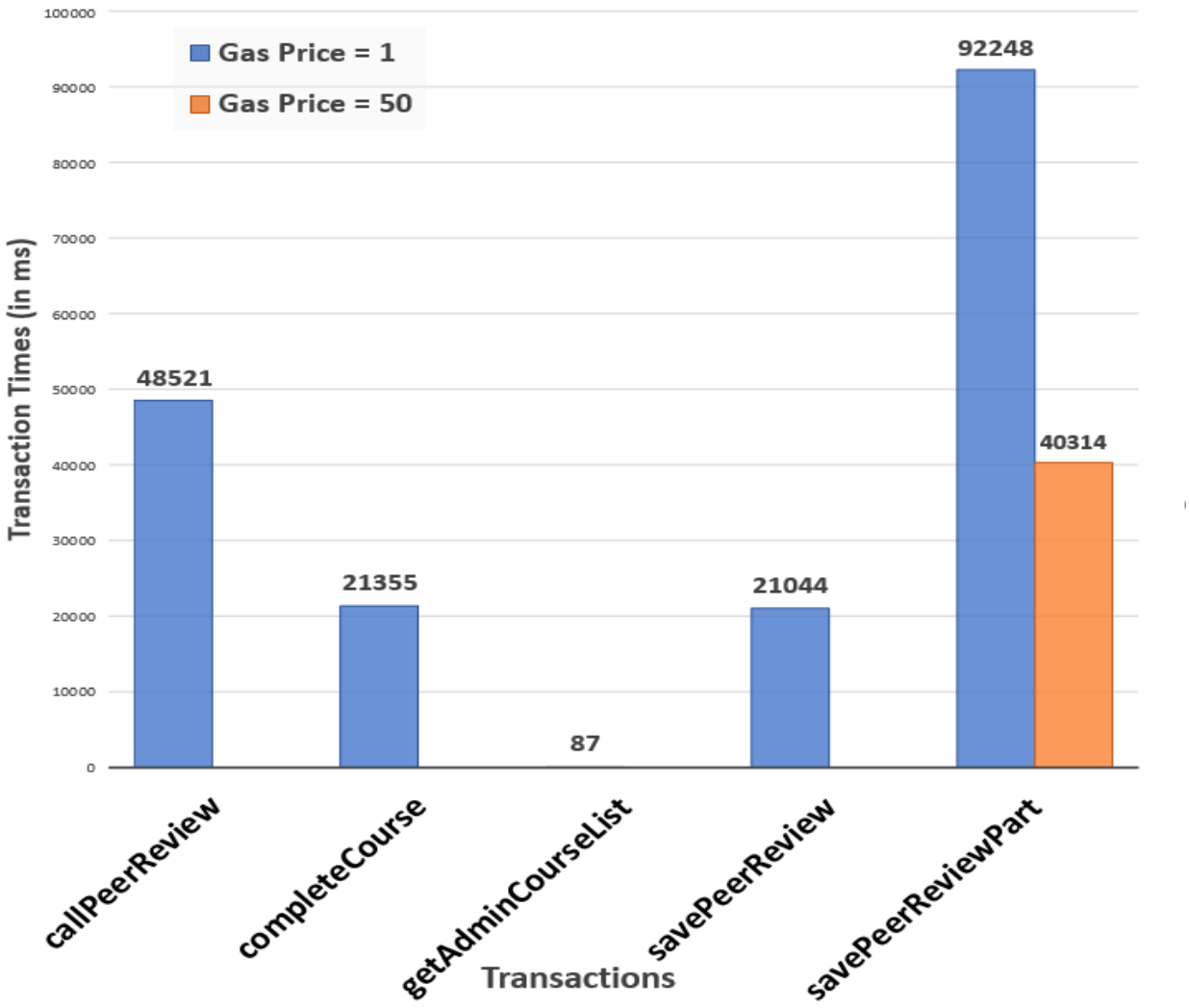

FIGURE 3.13. Peer Review Transaction Cost.
Note: The transaction times for calling different transactions (i.e. functions) during a Peer Review. This peer review model was simplified and does not yet calculate rigor, though requires manual entry from each peer for each part of an assignment. What was learned in this is that increasing the Gas Price increases the incentive, and subsequently transacts faster on the network.

It would especially establish how well it will be learned (i.e. the established rigor of the material taught). As students take these courses, potential employers will be able to connect with the system to help match what they are needing in a position with students who have proven successful in gaining that specific knowledge. This will help in the hiring process and may prove useful in finding and filling the many positions left unfilled due to a perceived skills-gap.

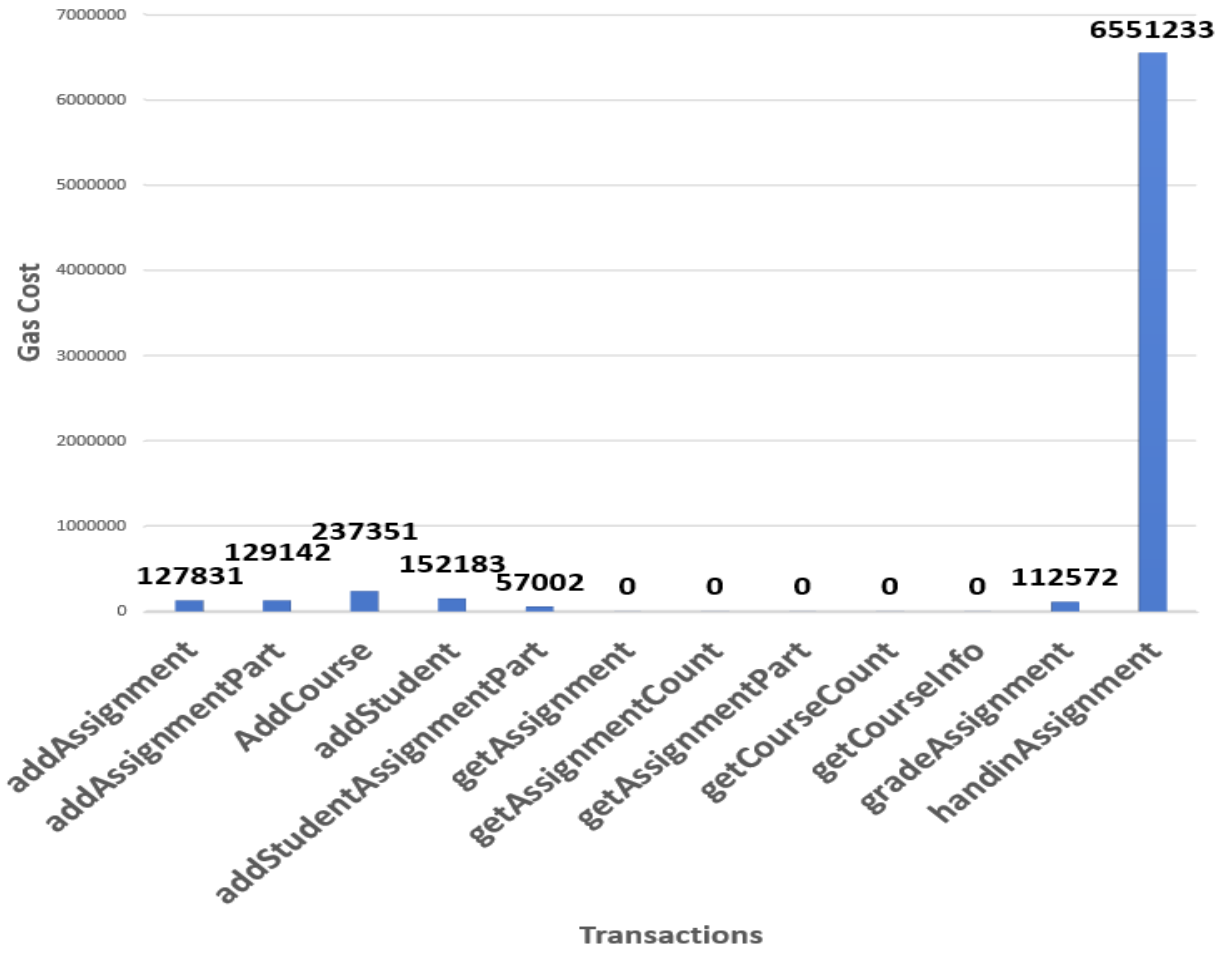

FIGURE 3.14. Gas Cost for Transactions by Student.
Note: This is the gas cost for each transaction related initiated by/for a Student. The transaction handinAssignment consumes the most gas due to the amount of work it is doing including parsing through the assignment – just turned in by Student – calling several internal functions including grading.

Tod Beardsley stated recently, "If you're only looking at college graduates with computer science or electrical engineering degrees from the top ten universities in the U.S. then yes, there are hardly any candidates, and most of them are going off to the five largest employers." [**45**] Our approach provides a possible way to open widely the pool of candidates to fill those positions by assuring their knowledge credentials, so employers can be better secured and trusting in understanding what a student has accurately learned, regardless of school reputation (i.e. does this student know, and how well, the required knowledge for a position).

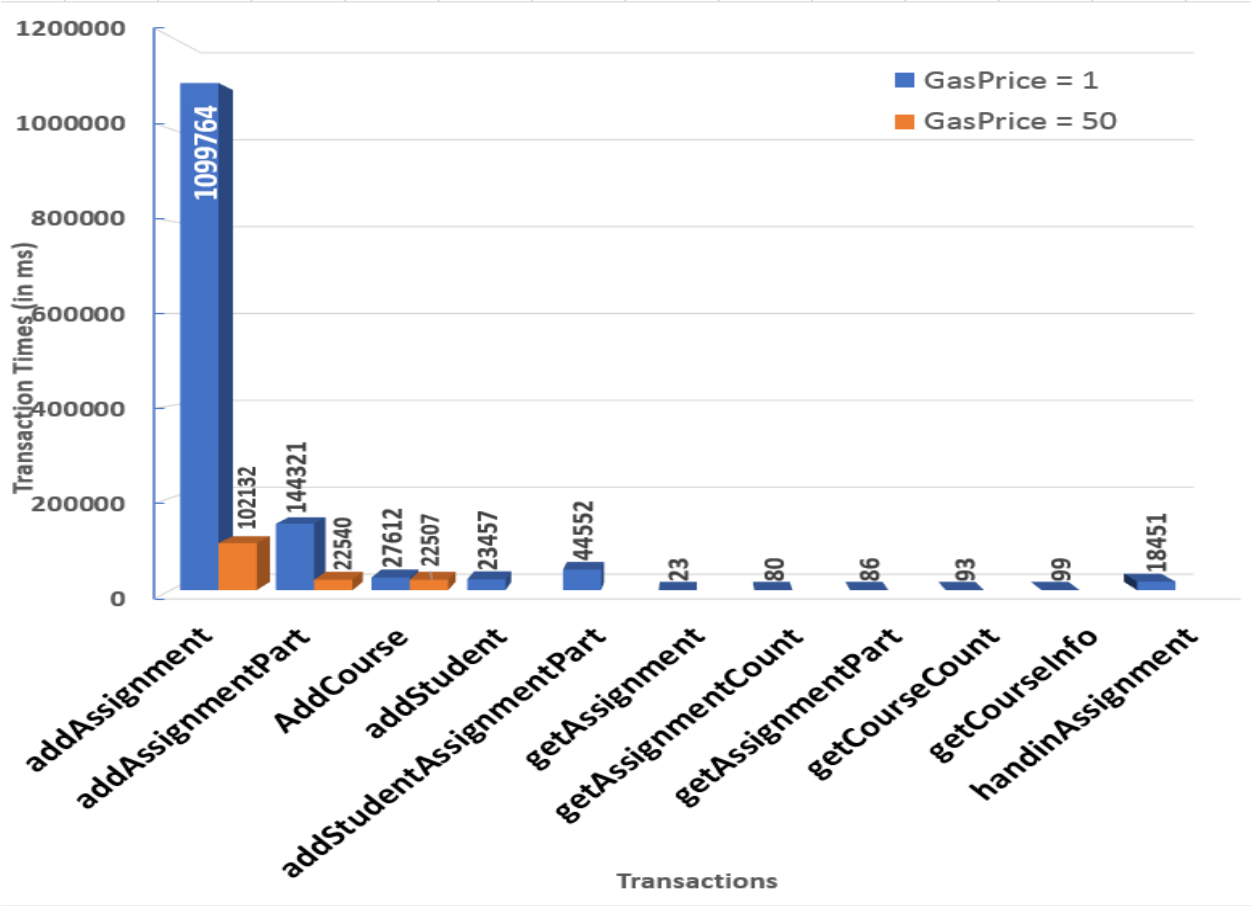

FIGURE 3.15. Transaction Times for Student Transactions.
Note: These are transactions to and for Students and their respective cost times in milliseconds to transact. Similar to (Fig. 3.13) for Gas Prices, when the Price was set higher, the miners were much more eager to do the work, and thus times were significantly faster. The addAssignment times dropped from approximately 18 minutes to 1.5 minutes. Though the handinAssignment costs the most to transact (Fig 3.14), it does the transaction in a reasonable time. More work is needed on transaction times overall.

3.7. Limitations

Limitations for this project in testing include the testing data sets utilized. As stated earlier, they are not a representative sample of the current state of hiring. In order to address this, future tests must include data sets that are taken from the company itself in order to properly model hiring processes. Another limitation in the data sets utilized is the rigor rating, as traditionally, that would be determined through ABET and CAE accreditation methods. However, these were taken through more research-oriented rigor rankings.

An important aspect that was left as future work is privacy. The system needs to protect assignment data from unauthorized user. Given FERPA requirements as well as organizations' user privacy policies, we need to prevent malicious players such as black-market dealers from accessing the course data. A standard combination of encryption and access control mechanisms should suffice to achieve these goals, but they are left out of scope of this work.

Finding a way to accommodate students with special needs is especially challenging using a blockchain. Smart contracts do not lend themselves well to multiple tracks of required coursework on the Etherium network [**72**]. We would need to find a way to accommodate special circumstances. (e.g. special needs kid requiring a shortened assignment, gifted child having additional work or more depth in the work, etc).

Lastly, though not implying this list is exhaustive, there are still logistics and scalability concerns. Can a blockchain handle the number of Teachers, Peers, Students, and Employers with all associated data in a feasible way (transaction cost and time)? Also, when running on Ropsten Testnet, probably due to the latency and long transaction speeds, our DApp would occassionally get confused and show assignments and assignment parts out of order. Even using promises and asyc/await clauses the course would show it out of order. This won't work well, since the handing in and grading depend on that order. It is possible that research is needed on "chunking" a blockchain consensus for massive-node systems.

3.8. Additional Considerations and Challenges

On the road to the defining our next problem, we focused on the issues currently associated with peer-reviewed course rigor and academic course equivalency for matching students' records between two schools (e.g., transfer students).

First, there is no peer-reviewed course rigor structure that is normalized across academic systems. The rigor of one course can be significantly different from a similar course at a different institution. Additionally, presently in peer reviews, whether of papers or courses, can have some elements where the reviewer can give a high quantitative review while commenting the opposite qualitatively (e.g., setting a 4 out of 5, and then incongru-

ously "trash-talking" the subject matter in the comments). Humans must weigh what they know of reviewer and their review against other relevant reviews and historical data. They come to a consensus on how to handle the internally-inconsistent review.

Currently, we are lacking regular, organized, and agreed-upon definitions of knowledge and skills, encompassing both academic and the workforce. This needs to change. A student's transcript of courses contains limited information. What a student learned in a course is not normalized from one institution to another, and rarely reflects what specific knowledge was learned. Comparing courses between schools is very challenging. Even when a general learned topic of a course is known, it is difficult to determine how well it was learned, as well as how rigorous that topic was covered.

The goal is to find solutions to solve the above problems. Moreover, we need to ensure consistency in how we define the knowledge learned across different institutions and industries. It is also important that the student's information is protected and secured. Definitively, students' credentials must be assured in a perpetually existing environment, and accessible by permission (i.e., on a "need to see" basis by allowed entities).

In this part of the research, we further explored using a blockchain to help solve these problems. Specifically, we at first honed in on the Hyperledger (HL) platform, instead of the Etherium network, Truffle platform, and Solidity language for smart contracts and processing.

At first we studied which HL framework is best for this project (e.g., Burrow, Fabric, Grid, Indy, Iroha, or Sawtooth). We decided we would use Caliper to collect metrics, Composer to help create and deploy contracts, and most likely HL Explorer to view transaction data. Also, we would need to either set up a Hyperledger Associate Member account for research (free), or use the Open Source Hyperledger, or go a completely different direction with another OS Blockchain network. Additionally, setting up our own network was important to test the various consensus algorithms and compare metrics. Eventually, we decided to move away from HL because we needed the ability to change out (or extend) the consensus model. We explored the Corda, Ouroboros, and Algorand platforms. For the next problem's

solution (Chapter 4), we selected Algorand since it was open source and its consensus model was easy to adapt and extend for our purposes.

3.9. Conclusions

From the results presented, we can deduce that the system was efficient. This is seen in the second metric, as all transactions with exception of one transaction, fell beneath a safe threshold. The exception transaction can be fixed in that through implementation of the proposed system, it is possible for one to further optimize the system.

We can also see that the system is scalable. Again, all transactions had O(1) complexity when varied with network load, except for credit. The scalability of the entire system, with all parts together, suggests possibly needing a different blockchain network requiring the consumption of fewer resources with faster transaction times. Since the cost to incentivize faster transaction times still leaves us with an undesirably slow process, new forms of consensus also may be needed to aid these above goals. The Ethereum network, currently, is not suitable for our future work needs. Although Ethereum developers are working on making that network more efficient in the future by introducing a type of ledger sharding and working towards a Proof of Stake consensus (versus Proof of Work that it presently uses) [**46**]. Since Ethereum 2.0, named Serenity, is not fully available or actualized, and we are needing to develop our own consensus model for Peer Review, moving to another open source blockchain model is necessary.

Finally, a can deduce your system is successful (partially). Outside the above scalability issues, success has been demonstrated through our earlier investigation of hiring rates from UT Austin and UC Berkeley, as the relative hiring ratios for the two institutions from the data given was quite close together. However, the algorithm places quite a bit of weight on rigor, which requires further investigation.

3.10. Future Work

Future work includes full implementation of the system by further fine-tuning the algorithm to properly account for rigor scores. This will allow us to further investigate the

relationship between rigor of an institution and hiring rates, and further optimize the dApp.

Part of our micro-accreditation research includes using a peer review system to determine the rigor of a course, then coming to a consensus on those review scores. Rigor is defined as the level of difficulty some problem is in regards to the expected level of understanding of the problem-solver. If it is more difficult, it has greater rigor. Applying rigor to topics in a course, we can better understand what a student has learned. Additionally, we can better compare what is learned from one institution to another. Peer-reviewed rigor scoring is very important for the accuracy of evaluating a student's knowledge. One example would be comparing a Calculus I class from Ivy Tech to one at Indiana University. As we currently do, we can make assumptions about the quality and rigor of what is learned. Assumptions are prone to human bias and misunderstandings, so we need to explore ways to prevent that.

Since we needed to trust that the peer reviews were honest, we needed a completely new automated reputation system. This led to exploring decentralized reputation management. The reputation of the peers providing rigor scores needs to come into the calculation for an overall rigor for a course, its topics, and its tasks. Meaning, those with a higher reputation have more influence on the total score.

How is a peer's reputation determined, gained, or lost? With that, how can we trust the reputation to be credible and not manipulated by malicious players and actions?

Our research evolved into an investigation and development of a system that also included both reputation and evaluation systems (discussed in Chapter 4). We figured we would eventually need a new consensus protocol to ensure a trust in that reputation system (discussed later in Chapter 5), but focused on answering the former question first. Since the micro-accreditation system is complex (e.g. using smart contracts to manage other smart contracts, etc.), we chose a simpler existing system instead, a decentralized marketplace, to implement and show how to answer this problem. Additionally, it allowed us to explore a non-PoW platform, which would fit better for micro-accreditation in the future. We did not have to worry about gas costs and other computational pains associated with PoW, and just

focus on solving our next problem.

CHAPTER 4

MAINTAINING REVIEW CREDIBILITY USING NLP, REPUTATION, AND BLOCKCHAIN

4.1. Introduction

Trust is one of the major concerns in any online marketplace framework. Major challenges in this setting include data privacy, trustworthiness, efficiency, and many other aspects which concern both buyers and sellers alike. In practice, all the parties with an interest in the marketing process may deviate from the prescribed procedures at any given point. Therefore, it is important to devise a mechanism that would allow us to gauge the parties' trustworthiness.

Customer feedback is a powerful tool, which is commonly used in trust management. Often, it is implemented as review systems. In many marketplaces, this feedback mechanism is recursive in the sense that it allows others to up- or down-vote reviews seen as helpful, that is to provide feedback on reviews. Some marketplace platforms assign a certain status to a party in order to enhance the credibility of their feedback. An example of this is Amazon's "verified buyer" status.

Other platforms, such as eBay, use human-driven reputation systems to ensure some level of trust in either the seller or buyer [**59**]. This time-consuming process is typically prone to biases and errors, in part due to the massive quantity of data to be evaluated. Note that it is still up to the buyer to evaluate products and sellers by manually going through the reviews and evaluating their accuracy—this adds an extra hurdle, now on the consumer's part. In practice, buyers usually rely on the top-rated (perceived as most helpful) or the most recent comments—which, in general, may not paint an objective picture. A natural question, which arises in this context, is what kind of mechanism may enable us to deliver an objective view of a product to the customer. In particular, we focus on the setting of decentralized marketplaces, which have been gaining popularity in recent years, such as BitBay, OpenSea, Ocean Market, Origin Protocol Markets, to mention a few.

It is worth noting that trust in the context of online marketplaces has a wide variety of aspects. For instance, the buyer trusts the seller to accurately present the product, to deliver it in a timely fashion, to properly process payments and reimbursements, and so on. These aspects have been addressed in a large body of literature, such as, e.g., [**13**, **57**].

In this work, we are focusing on a particular aspect of trust in reviews and reputation of the parties who provide them: review credibility. Specifically, we are focusing on the following particular scenario, which is the first step towards the above-mentioned mechanism: A party leaves a review (say, on a product or service), which consists of a text and a rating. We will use NLP to evaluate the "positivity" of the text, and then we will compare it to the rating. A trustworthy review is expected to have a good match of the positivity to the rating.

Next, we need a system that would provide an incentive for the reviewers' trustworthy behavior. A natural solution would be to apply a reputation system in order to leverage trust in the reviewers and their feedback. A moderator could be hired to handle incongruent ratings, as well as spam and fake reviews. They would be charged with processing reputation adjustments and dolling penalties for infractions. The challenges with this are the substantial human resources needed in both cost and time due to the manual handling, and being prone to bias analysis due to human handling. A better approach is to handle the review evaluation automatically using NLP, with the reputation and rating administration also driven automatically. Finally, in the decentralized scenario, it is natural to use blockchain for storing the reputation values and updating them (automatically, using NLP evaluation) according to the reviewers' performance

The work presented in this chapter was published as "Maintaining Review Credibility Using NLP, Reputation, and Blockchain" in the 2022 IEEE 4th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications Proceedings [**74**].

4.2. Related Works

There is a large body of work on applications of blockchain to decentralized marketplaces, e.g., [**9**, **20**, **22**, **33**, **34**, **55**], to mention just a few. Let us briefly discuss the works,

which are the most relevant to our result.

Bajoudah et al. [**9**] apply a trust and reputation framework on top of a marketplace model to encourage trustworthy sellers. The paper suggests a model where an increase in reputation (which is not compared against others in detail) decreases the cost of the overall data delivery. This is achieved by reducing the number of transactional checkpoints that are ensuring an honest delivery. A low reputation would require more checkpoints than a trusted high reputation would, therefore incurring higher overall transaction costs. Although their reputation model is similar to ours in that it is pervasive through each transaction and block, the reputation is stored and used at the dApp level, instead of in the core. Though in our model, reputation can be used for purposes at the dApp level, one of our main focuses is using reputation as part of the consensus process at the core. Also, the authors do not specify how the reputation model functions, that is, they explain that the reputation model used is currently outside the scope of the paper, leaving it to future work.

Jiang et al. [**33**] show a correlation with product ratings, review texts, and customers' probability of purchasing products through the use of sentimental analysis (SA) to create a "product reputation". They use SA as we do, however, they assign a reputation to a product, rather than to a reviewer, as in our case.

Joshi and Kumar [**34**] introduce a reputation and blockchain model to keep fraudulent reviewers and sellers in check. Specifically, they set some initial requirements to be a reviewer, and they limit the seller's control over the final price of their product. The reviewers are submitting estimated prices against the seller's price rather than feedback on the product. Although this paper introduces an approach that is somewhat similar to ours in using weighted reputation, their model is mainly focused on marketing, such as determining accurate product cost estimates. On the contrary, our proposal focuses on an accurate evaluation of the reviewers' reputations.

Salau et al. [**60**] use reputation as a stake in the context of data cooperatives [**29**]. The idea of using reputation to incentivize the parties' honest behavior—which originates from earlier works such as [**26**]—is employed in our construction as well.

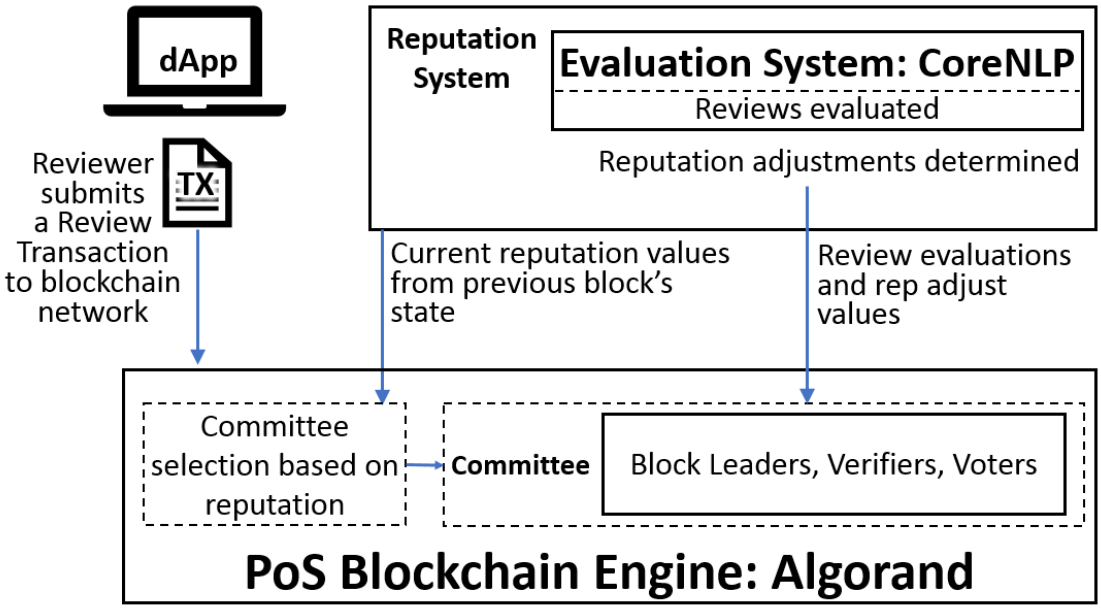

FIGURE 4.1. High-Level Architecture of Our Proposal.
Note: The proposed model contains an NLP evaluation system (CoreNLP) and a custom reputation system, where the reputation values are stored on a Proof-of-Stake blockchain (Algorand).

However, we apply it in a different context of evaluating review credibility. Also, the work [**60**] uses a different blockchain system, the Snow White protocol.

4.3. Overview of Our Contribution

We propose an architecture for the reputation-based review evaluation system, which is built on top of the blockchain system, in order to ensure correct and trustworthy assessments. In our proposal, the trustworthiness of reviews is evaluated using NLP and the reviewers are assigned a reputation according to this evaluation. The reputation is stored on the blockchain and is used as an asset for the consensus process.

We constructed a proof-of-concept implementation of the proposed architecture, which is shown in Fig. 4.1. We chose Algorand as the underlying blockchain system, because it was easy to adapt its committee selection mechanism for our purposes.

In a marketplace, we consider sellers to be more regularly and consistently available online than buyers. Additionally, we consider buyers to be more active in submitting reviews, consequently reflecting a longer, more credible history of reputation. For this reason, we

suppose that the pool of nodes (we will also refer to them as "players") will be restricted to buyers. Recall that the nodes will take part in the consensus protocol, hence running the blockchain system. At each step in a round of the consensus protocol, a new, anonymous, and random set of players is selected to form a committee. In our approach, for the reasons stated above, Algorand nodes are now directed to only consider the sellers' reputations in self-selecting for a committee.

We consider the following three main aspects of assessments made by the parties (sellers and buyers) involved in the marketplace. The reputation of the seller, that of the buyer, and the rating of the product or service offered (henceforth referred to as "Product", for simplicity). The sellers' reputation values are derived from reviews submitted by buyers, and in return the sellers review buyers (e.g., their speed of payment, quality of communication, and other points). The buyers also review Products they have purchased (henceforth, we called it a "Rating" which is given to a Product). As discussed later, in general, the reputation of the parties is adjusted positively or negatively in each round based on the comparative evaluation of the reviews and ratings they submit in each round.

Reviewers submit their reviews as transactions in the blockchain system. A chat dApp was created for this purpose, where any message from a user (that is a reviewer) is interpreted as a review. A reputation system is implemented as a stored numerical value associated with each reviewer's public key, and stored on the blockchain as part of the current state of the system in that round's block. We note that the reviewer is in effect a user in the Algorand system. This means that a single user may create multiple identities if they wish. However, since a beginner is assigned a minimal reputation of zero, it is beneficial for the user to work under the same identity in our system, in order to build up their reputation.

Then, the reviewers' reputation is adjusted in each round, according to the NLP-based evaluation. This evaluation system analyzes the sentiment (negative, neutral, positive) of a review text to determine congruency with the associated submitted review rating. As mentioned, the reviewer's reputation is adjusted based on this analysis, and the state of all reputation values is stored on the blockchain at the end of each round.

By mapping reputation to stake, these reputation values assume the role of tokens. Hence, the reputation forms a "quasi-stake", which each reviewer has in the system. This approach is the same way as it is done, e.g., in [**60**]. This way, the reputation is used as an asset/incentive to encourage honest behavior of the participants.

Our simulation results showed that the NLP component incurred a reasonably small delay to review transactions. Specifically, we compare the time requires to add a standard payment transaction in Algorand with that to add our (custom-designed) review transaction, and for about 50% of them the timing was similar, while most of the outliers took about 2-3 times longer. Also, we observed that the NLP component ensures an accurate credible evaluation of the product review texts taken from a custom-picked dataset of Amazon reviews.

In our work, we augment the Algorand protocol (refer to 2.4 to include both an NLP-based evaluation system for reviews and a reputation system reflecting the peers' ranking (see next sections).

4.4. Review Analysis and Reputation Management

4.4.1. Review Analysis Using NLP

Our proposed architecture includes the NLP component, which is used for the analysis of the reviews. Specifically, we apply sentimental analysis (SA) [**66**], which is typically used for data analysis in various areas, such as social media [**21**] and health care [**56**, **68**], to mention a few. We refer the reader to the survey [**28**] by Gupta et al. on details of this topic.

SA is the process of breaking down sentences and paragraphs into finite terms to identify the writer's feelings in relation to the categorized bounds of positive, negative, or neutral attitudes towards a specific topic. In our design, we use SA to gauge the "accuracy" of the customer's textual review with their numerical rating (e.g., "4 out of 5 stars"). This accuracy directly affects the adjustments to a reviewer's reputation.

SA is used in several NLP tools, such as Lexalytics [**40**], Google Cloud Natural Language [**27**], and Natural Language Toolkit (NLTK for python) [**52**]. We chose to use

the Stanford CoreNLP [**47**] suite of tools, because it is a comprehensive "out of the box" ready-to-use open-source system that fits our purposes. Their recursive neural network model is already trained on a sentiment dataset for movie reviews, hence it fits well for our evaluation system needs. Another useful feature of CoreNLP is that it furnishes instructions for extending and training their model further, and it maintains online support.

Specifically, CoreNLP calculates the positivity of the text on a 0-100 scale (100 being the most positive). In other words, the sentimental analysis determines how positive the reviewer is expressing themselves, converging the result into a numeric scale. Though SA can be seen as a mechanism to determine the degree of polarity of both positive and negative in the opinion, we focus on positivity.

Then, accuracy is ascertained by comparing the evaluation score output by CoreNLP to the review rating scaled to the range [0, 100]. Specifically, in our application, we first map the original quantitative (review rating) interval [a, b] to [0, 100] to more easily compare against the qualitative (review note) interval of [0, 100]. (See Table 4.1 for the structure of a Review Transaction).

The scaled quantitative (rating) value is computed as follows:

$$f = \frac{(100 - 0)}{(b - a)} v \equiv \frac{100}{(b - a)} v,$$

where v is an original quantitative (rating) value.

Example: With a rating system of [0, 5], the scaled value is computed as $f = 20v$.

Denote the NLP qualitative score for a particular review as $Q \in [0, 100]$. We can compare it to f and use the discrepancy to determine the adjustment to the reviewer's reputation. The details of how the adjustment is done are discussed in the next subsection.

Example: Suppose that the book review reads "This is very good!", and a given rating is 3 out of 5 stars. Also, suppose that the NLP system evaluates this phrase as 80% positive ($Q = 80$). Then, the scaled rating is $f = 20 \cdot 3 = 60$.

TABLE 4.1. Fields of a Review Transaction.

Field	Type	Default Value
Type	Tx Type	Review Tx
Header	Header Type	(transaction header info)
ReviewNote	byte[]	empty (text-based review)
ReviewRate	numerical	0 (quantitative rating)
ReviewEval	numerical	empty (scale 0-100)
RepAdjust	numerical	0

Note: We note that in the actual Review Transaction, there are other fields, which are either blockchain-related data for transactions (such as, e.g., sender, receiver, etc.) or others that are not used in our design (such as, e.g., fees, notes, etc.). They are omitted from this table for the sake of simplicity.

4.4.2. Reputation Systems

Reputation systems have a long history of applications both in computer networks and also in blockchain systems [**6**, **8**, **14**]. In our proof-of-concept implementation, we use a simple variation of the AIDM algorithm [**17**], which is described below and with further specifics in the next section (High-Level Architecture).

Each user has a reputation value $\mathcal{RV}$ also associated with their public key (PK) in our system. The $\mathcal{RV}$ can be used in evaluation of users, services, and products. In our case, we are using it specifically with users. Beyond this, $\mathcal{RV}$ can be used to calculate a weighted overall rating for some product or service. Also, it can serve in the selection process of committees each round by using that reputation in calculating a pseudo-stake, instead of tokens (money).

Reputation is earned following a common policy of additive-increase, meaning incrementally one unit at a time. Using the multiplicative decrease as punishment reduces the player's reputation—and subsequently their weight—quite quickly. Therefore, this mitigates any intended payload damage, e.g., the use of a high-reputation / high-stake position to increase the probability for contiguous successful malicious actions. The integration of this algorithm is discussed in Section 4.5.3.

4.5. Proposed Architecture

In this section, we describe the proposed system in detail. Specifically, we discuss the changes, which we made to the Algorand protocol, in order to use the consensus function with reputation instead of stake.

4.5.1. High-Level Architecture

As shown in Fig. 4.1, as the underlying blockchain system, we use Algorand. We extend a blockchain's consensus protocol model to include both a reputation system and a review evaluation mechanism. The latter is responsible for the reputation adjustment.

Some main modifications of Algorand include adding a new reputation field and a new type of transaction called a Review Transaction. We then changed how stake was used through mapping this new reputation to stake. This furthermore altered how a committee is formed from this new mapping. Additionally, some steps of the consensus protocol were modified, introducing a new activity for leaders and verifiers in evaluating reviews. Based upon these evaluations, we devised a system to adjust the reputations of the reviewers, and now store these participants' reputations as part of the current state of the system in a round's block.

4.5.2. Modification of Algorand

We have added a new reputation field which is associated with each player's public key. With its numeric value starting at zero, it is codified to not allow negative values, and have a maximum value of 10000, an arbitrary number selected to limit the field to prevent possible integer overflows if unrestrained. As mentioned above, we map this reputation to stake, with these reputation values assuming the role of tokens. This forms a quasi-stake, which each player has in the system.

Furthermore, we modify how committee participants are self-selected in each round. Instead of employing the number of tokens a player has in the input to Algorand's verifiable random function, we apply the player's reputation value. The higher the reputation the greater the chance of being selected for a committee.

In each round, the reputation values are received from the reputation system from the blockchain's last block state. Any changes to that state are stored in the current block at the end of the current round.

Additionally, we added a new type of transaction named Review Transaction, which does not require payment—see Table 4.1. This type of transaction contains a review text, a review rating, an evaluation rating, and a reputation adjustment value. Potential Leaders are now tasked to call for evaluating these Review Transactions, and Verifiers later analyze those evaluations similarly as part of their validations. A review is evaluated for congruency (e.g., checking for fake inconsistent reviews) in addition to other deviant behavior such as spamming and showing bias.

Before the round ends, we have introduced an algorithm that calculates a reputation adjustment based on an accepted Leader's evaluation. Then, another function subsequently adjusts that reviewer's reputation, which consequently could affect a user's "stake". For this adjustment, we employ the additive increase and multiplicative decrease method in our algorithm, which is discussed explicitly in the next section.

Our approach does not allow trading, buying, or selling of reputation; therefore, changes to reputation cannot be traced back through previous transactions between different parties. For review transactions, we removed the need for a second party. We can verify a party's reputation value by tracing the user's review transactions and associated adjustment fields stored on the blockchain. Recall that each review transaction stored in a block on the blockchain contains the review itself, the NLP-evaluated score, and the $\mathcal{RV}$ adjustment made. This furnishes the tools needed to validate those values by tracing the transactions that changed the $\mathcal{RV}$ of a party over time.

4.5.3. Integrating the Review Evaluation Component

Our review evaluation component consists of the reputation system and the NLP evaluation system (cf. Fig. 4.1).

Each (honest) reviewer has a reputation $\mathcal{RV}$ assigned to them, with their reputation being adjusted only when all the below conditions hold:

- The reviewer i had submitted a review (for simplicity, we assume that a numerical rating is a part of it);
- A Review Transaction was created and it was included in a block, which was added to the blockchain in the current round.

For purposes of this discussion and to simplify the concept, let us constrain the player's maliciousness to only incongruent reviews, and start the $\mathcal{RV}$ at 0. Also, we will restrict a reviewer to submit only one review per round, so that the latter is added to a single block in some round.

Obviously, one may define different rules for adjusting the reputation, which will depend on a particular application. In our work, for simplicity, we use the absolute value of a difference between the NLP qualitative score Q and the scaled quantitative rating f. This quantity is denoted as $\Delta = |Q - f|$. Now, for our testing, we set the rule that if $\Delta \leq 20$, the rating is adequate and the reviewer's reputation increases, and otherwise it decreases. We choose this rule for simplicity of our proof-of-concept implementation. In principle, different rules can be used depending on a specific application.

Taking the above rule into account, we use a variation of an AIMD algorithm [**17**] to compute the reputation adjustment as follows:

$$\mathcal{RV}_i(r+1) = \begin{cases} \mathcal{RV}_i(r) + \delta, & \text{if } \Delta \leq 20 \\ \mathcal{RV}_i(r) * \lambda, & \text{if } \Delta > 20 \end{cases}$$

where r is the current round, $\mathcal{RV}_i(r)$ is i-th user's reputation in round r, δ is the additive increase parameter ($\delta > 0$), and λ is the multiplicative decrease parameter ($0 < \lambda < 1$). In our simulations, for simplicity, we set $\delta = 1$ and $\lambda = 0.5$.

We explored other reputation adjustment methods, but they didn't fit well for our purpose. Consider the simple AIAD, where reputation increases and decreases in steps. This would allow "bad" actors to continue much longer in the system. Comparatively, AIMD swiftly reduces the player's reputation and effectiveness in the system.

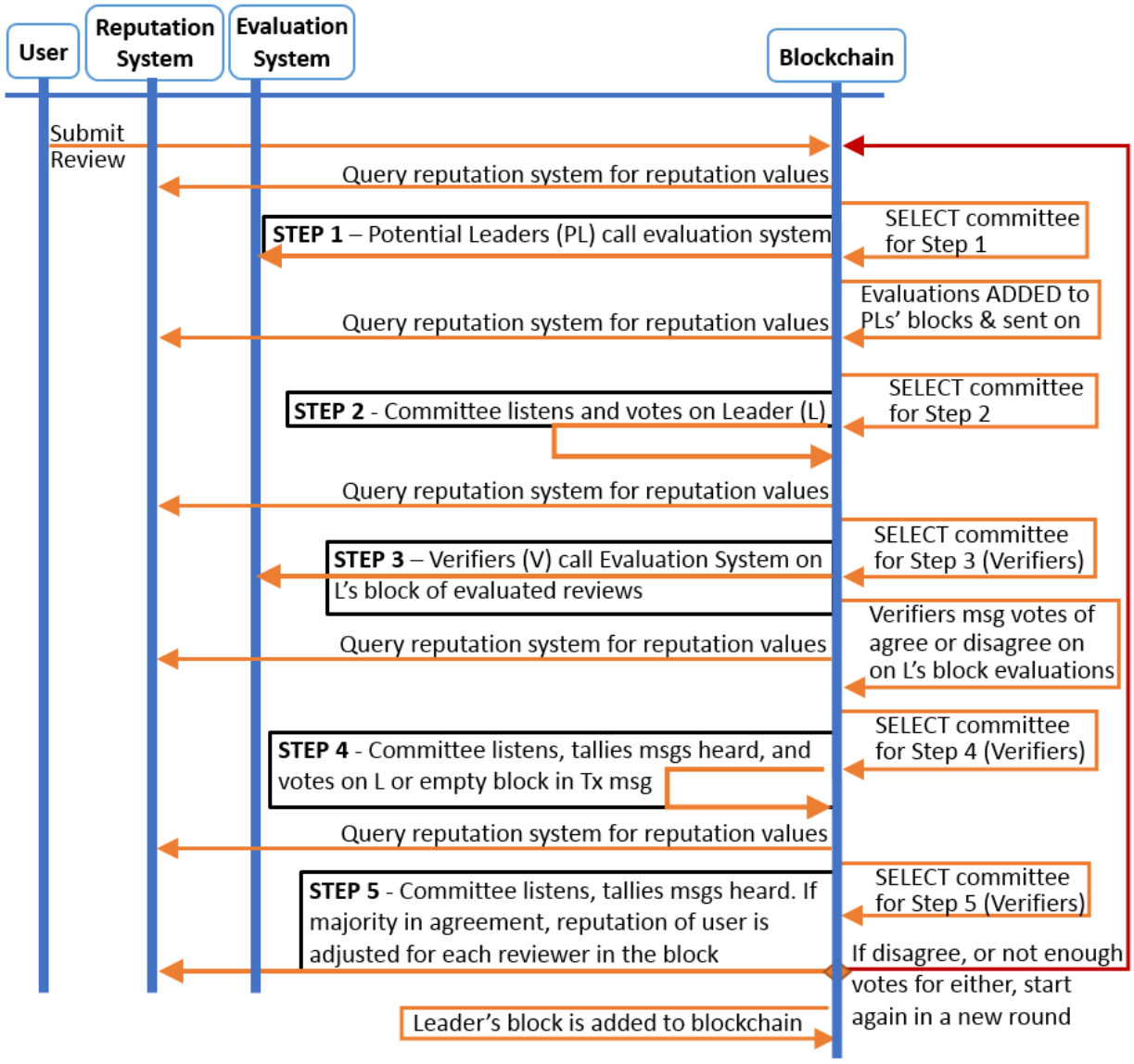

FIGURE 4.2. Workflow of the Proposed System.
Note: A review is submitted to the Blockchain nodes' transaction pool. This shows the set of interactions with the PoS Blockchain, Reputation-based system, and an Evaluation system.

4.5.4. Modifying the Algorand Core Code

As mentioned earlier, our implementation for testing uses a derivative of the Algorand blockchain core code. To simplify our description, we will limit our discussion to review-related notions and use Algorand's steps for convenience in our explanations. We present a general workflow overview of these steps in Fig. 4.2 and more specifically below. To start,

a reviewer using a dApp submits a review, creating a review transaction. At every step, verifiers are selected randomly to form a committee (see the previous section on Modification of Algorand) for verifying, voting, or evaluating the reviews and are anonymous to each other. This process follows these steps:

- **Step 1:** In addition to Algorand's standard security checks (e.g., valid signatures, etc.), potential leaders evaluate each review transaction by calling the NLP evaluation system to retrieve a review score and reputation adjustment values before adding it to their potential block. Any invalid reviews are removed.
- **Step 2:** A new group of randomly selected participants agree on the leader in the same way as in Algorand.
- **Step 3:** Verifiers "open" the leader's block and re-evaluate each review, comparing their evaluation to the leader's evaluation. If the results are matching, the original review's evaluation is considered good. When all the reviews are considered good, then the leader's block is deemed good. If there is more than one divergent evaluation, then the entire block is considered bad, and a vote of no confidence for the leader and a vote for an empty block is sent.
- **Step 4:** The fourth group of verifiers counts all the votes of the previous group. This step is done in the same manner as in Algorand.
- **Step 5:** The final step is the application of the reputation adjustments from each review transaction in the approved block (if there no approved block, this step is skipped). The block is added to the blockchain, with the new reputation values stored as part of the current state of the system in that round's block.

4.5.5. Analysis of the Proposed System

Let us discuss the functionality of our system. Even after the modifications, the system works. Adding reputation values and mapping them to stake does not change the way the underlying blockchain system functions. Committee selection and the consensus on a block are still handled in a similar way. We assume that automated adjustments to a player's

reputation will either positively or negatively affect their probability to be selected to engage in the consensus process. We expect the system to deliver the performace comparable to that of Algorand where players buy or sell their tokens. Although CoreNLP adds an average 36 seconds delay due to processing time to evaluate a review, we expect the system to come to a consensus on a block within a reasonable time frame.

There could be a review transaction created but not added to a block. We assume the transaction is signed correctly and the player is allowed to submit a review as a transaction to the system. In cases where it is not, we expect the review transaction to be handled in the same way as the blockchain does by removing it from the transaction pool. The system will never consider that review again.

If a reviewer creates a review, but somehow it never gets added to a block due to running out of time repeatedly, the transaction will expire. An expired transaction is removed from consideration and no further action is performed on it. This is the way Algorand handles the similar cases, so we expect the system to continue working in the regular manner. In our model, the review will cease to exist and reputation will not be adjusted. This does not affect the blockchain.

4.6. Simulation Results

4.6.1. Testbed

In order to test our proof of concept, we developed the dApp, which is a modified chat application written in the programming language Go. We augmented and adapted the code in this dApp, the associated SDK, and the main core blockchain system (Algorand) to test our new system. To the dApp, we fed the data (reviews) both manually and through using a file loaded at run-time.

Since decentralized marketplaces are still relatively new, there are not many publicly available datasets. For this reason, we used data from Amazon. Specifically, we handpicked random product reviews, which were relatively short, 2-3 sentences maximum, and placed them into a JSON file to be loaded to the dApp. For testing, we read this dataset file, or manually entered data into the dApp input box (especially for the transaction timing tests).

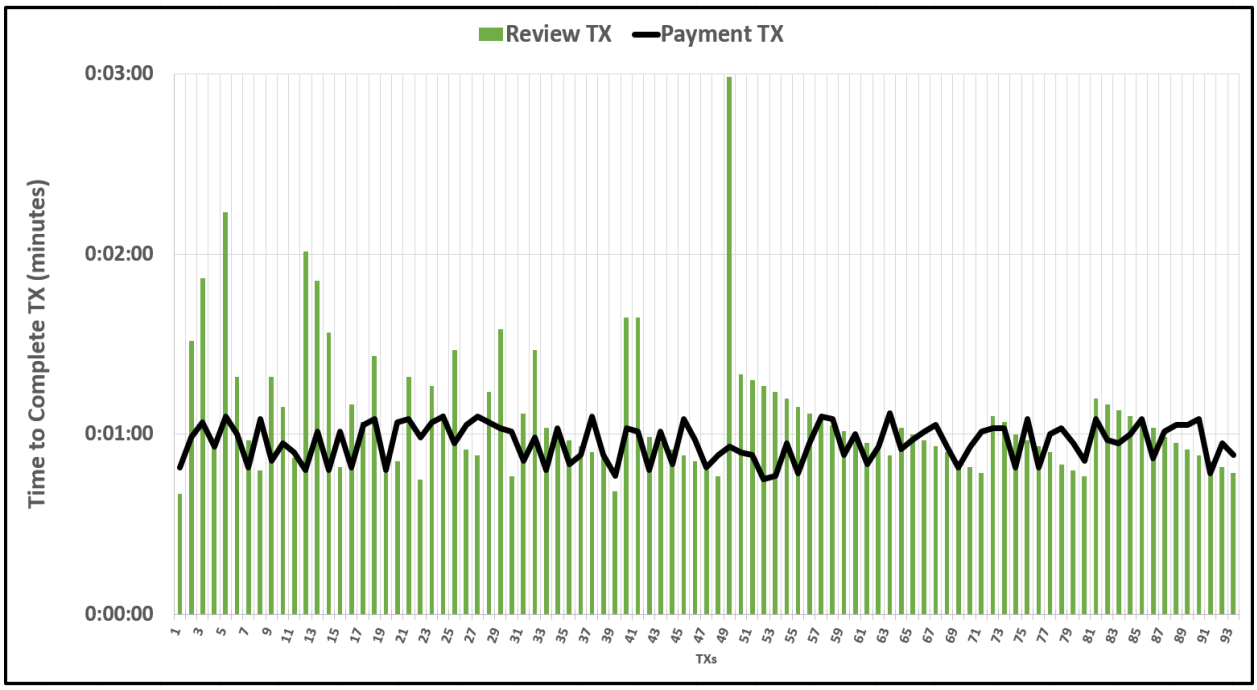

FIGURE 4.3. Consensus Timings for Payment and Review Transactions. Note: On average, the Review transactions are 9 seconds slower as compared to the Payment transactions.

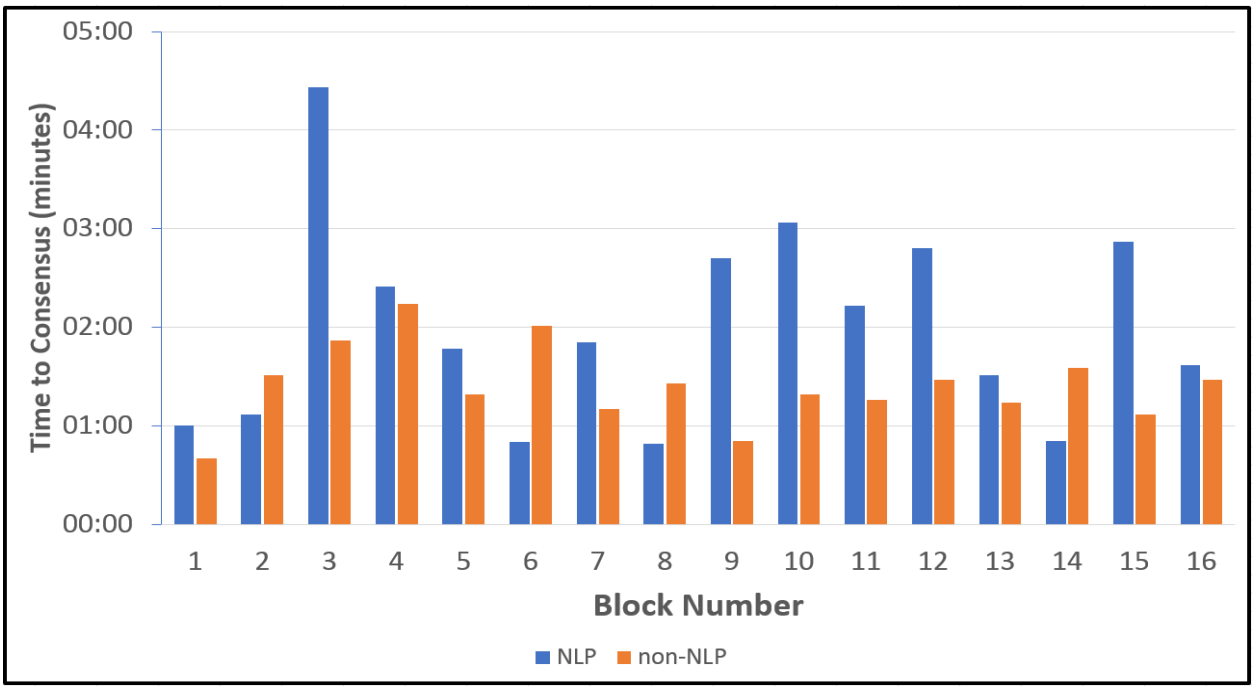

FIGURE 4.4. NLP Evaluation Overhead for Block Consensus Timing. Note: The block consensus timings are shown for the case when the real-time NLP evaluation is used versus when it is not used. The average delay introduced by the NLP component is 36 seconds.

All our test were made on a single Ubuntu 18.04 Linux system running on a Dell Optiplex 7040 8-core, with 16GB memory (henceforth, we call it "the system").

The implementation starts with installing AlgorandPoR [**2**] ("core"), the SDK [**3**], and downloading the proof of concept dApp (algochatPoR) [**1**] on the system, following instructions found in the source. In the PoC dApp, we used Python scripts to create and launch a private test network specified in a configurable template. Creating a network calls the core to setup nodes with associated wallets and other blockchain parameters including the network name. Once that is complete, the scripts create each node's directory structure and startup shell scripts. Once finished, we call the script to start the network.

Our most common system setup started with three simulated nodes (clients), each in their own console window tab. In each tab, an instance of AlgoChatPoR is launched, presenting a console user interface (CUI) allowing a user to enter text to submit a payment or a review transaction with the help of the SDK. To load the JSON data file of reviews, we had to quit one node's instance, then relaunch it with its associated generated shell script and some command-line arguments (e.g. "-autofile ./data.json") to specify which file to load. The SDK defines what a review transaction is, wrapping the submitted review and rating into a newly created transaction, and finally broadcasting that to the network. AlgorandPoR core installation is coded to handle that transaction (see the steps of consensus in a previous section).

4.6.2. Metrics and Simulation Results

Our first focus was testing transaction times (or how long it takes for a transaction to be added to a block). In other words, we were investigating whether review transactions are slower than the regular payment transactions. Next, we ran a study on an overhead which the NLP component introduces. These tests are important for showing that the new review transaction does not affect the timing significantly, and that the system continues to perform well even when review transaction are evaluated by NLP in real time.

Next, we tested an accuracy of the NLP evaluation. We expect the NLP technology to perform similarly to what human would deliver— the results are shown in Fig. 4.5.

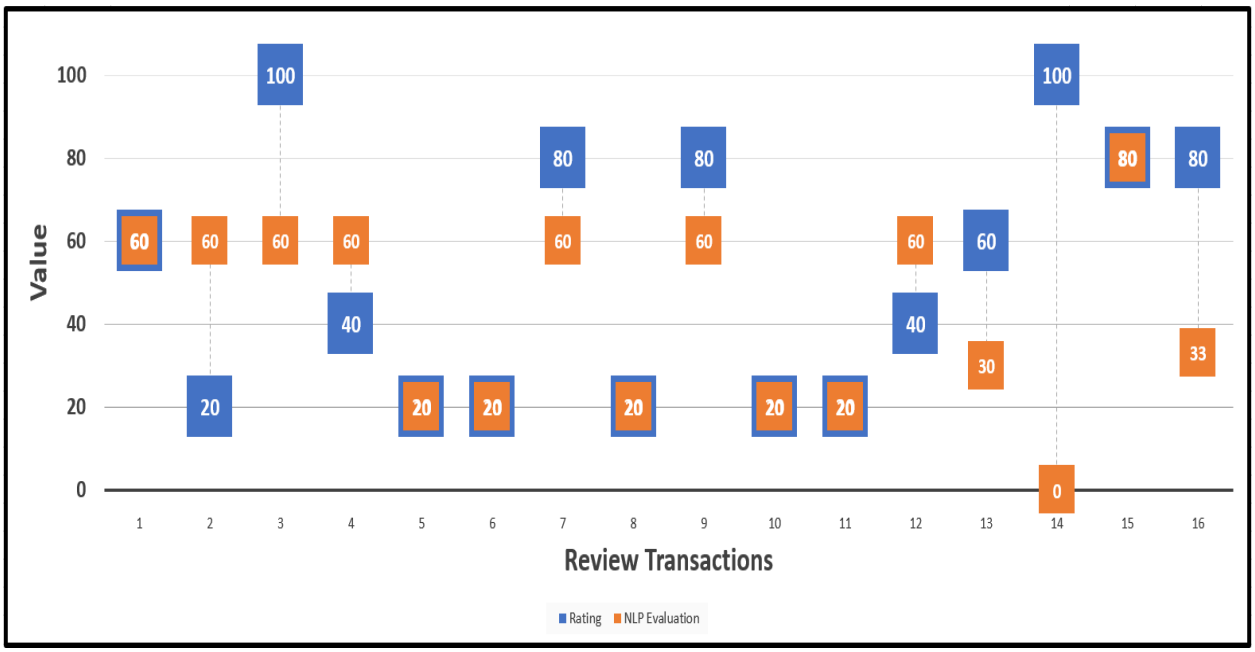

FIGURE 4.5. NLP Scores Versus Ground Truth (Human Evaluation).
Note: The average difference between the NLP evaluation and the ground truth is about 21 points, with the standard deviation of those differences at 26.

In Fig. 4.3, we compared consensus times for both Payment and Review transactions. We used a simulated non-live evaluation of reviews for the Review transaction. The block consensus times for review transactions range from 40 seconds to 1 minute 52 seconds, with a few outliers above 2 minutes, when using tokens as a stake type in a Proof of Stake system. The timings were similar for both types of transactions. This shows that using different transaction types in a non-live evaluation setting does not affect the overall time for consensus on a block to be added to the ledger.

In Fig. 4.4, we tested block consensus times when using real-time NLP to evaluate reviews using a small dataset adapted from Amazon's data (see Fig. 4.2). This dataset was handpicked from random product reviews, 2-3 sentences maximum, and placed into a JSON file to be loaded at dApp runtime. Since we are using NLP's resource-heavy sentimental analysis evaluation on three simulated nodes, we limit each block to one review transaction. The block consensus times for real-time NLP range from 49 seconds to 3 minutes and 4 seconds (with one outlier of 4 minutes 26 seconds).

TABLE 4.2. Sample Reviews and Ratings.

TX	REVIEW TEXT USED	RATING	SCALED
1	This is Good	3	60
2	Best book ever read	1	20
3	Best item ever and awesome fantastic	5	100
4	Some good and some bad it's ok	2	40
5	trash filled with more trash	1	20
6	I've never seen anything so bad and worthless	1	20
7	This is the best thing ever	4	80
8	I do not recommend! It doesn't work	1	20
9	good but has a lot of work to do to be great	4	80
10	if I could rate this 0, I would	1	20
11	This is so awesome that I wouldn't give it to my enemy	1	20
12	recommend to everyone. Best ever. Super	2	40
13	it's ok. won't buy again. likely will trash it. Yuck	3	60
14	total fake knockoff that uses junk material	5	100
15	This is very Good	4	80
16	good in the trash. great in the dump. buy it to junk it	4	80

Note: The NLP evaluations are the scaled values computed as described in Section 4.4.1.

This shows that time to consensus can fluctuate when using a real-time evaluation system, with more than 50% of the transactions finishing within the range of cases when NLP is not used (less than 1 minute 52 seconds). This may be caused by either the complexity of the text being evaluated in Fig. 4.5 or the resources available during the round. We emphasize that the custom-selected review dataset consisted of comments of about 2-3 sentences, and therefore the real-life timing may differ from the one reported above, depending on a specific dataset.

In Fig. 4.5, we tested the accuracy of the NLP evaluation feature. Our goal is to use the technology which evaluates reviews in the same way as humans do. This means that if a person perceives a review as mostly positive, then the NLP system should also appraise the same text as mostly positive. The same dataset of product reviews was used (Fig. 4.2) as in Fig. 4.4. We gauge how positive the review comment is through sentimental

analysis, which provides a single number (0-5) evaluation for each sentence, where 0 denotes very negative, and 5 denotes very positive. If this system is provided multiple sentences, it returns an evaluation for each sentence. Our implementation computes an average of all those evaluation scores and then the result is scaled to the invterval (0-100). In some cases, the rating matches the evaluation: e.g., see the review text 5, "Trash filled with more trash". In other cases, the rating given by the reviewer does not match the associated text: e.g., see the review text 2, "Best book ever read". In the latter case, the reviewer's rating does not match their comment; however, our system catches this, evaluates the comment accurately, and determines the level of (in)congruence. From this, our system will decide whether the reputation should be increased or decreased.

4.7. Conclusion

With this new model, we have moved towards showing how technology can be used to evaluate the trustworthiness of both the reviews and the corresponding reviewers. We deploy NLP to determine whether the reviews are congruent and trustworthy.

We note that this information can also be useful at a higher level, such as in dApps for various purposes. For instance, a dApp can rely on this immutable information, using the evaluation scores of the reviews for making decisions. The sole incentive in our model is to increase one's reputation. No financial incentive is currently present, although it may be introduced in the future if potential applications demand it. Our system provides an accurate way to implement an automated analysis of reviews ensuring the trustworthiness of the evaluation.

Our preliminary results show comparable block consensus timings for the cases of using tokens or reputation as a stake in our Proof of Stake component. Additionally, we show that a real-time NLP evaluation may introduce a substantial overhead to about 50% of transactiona. This may be caused by either the complexity of the text being evaluated or the resources available during that round.

More significantly, by using a dataset adapted from Amazon product reviews, it is demonstrated the NLP evaluation component performs similarly to a human evaluation

(ground truth). As mentioned in previous sections, our proposal derives the trustworthiness and credibility of a participant via evaluation of their reviews, which is in turn reflected in their reputation. Then, this reputation is used by the consensus algorithm.

Our future research focus is on constructing a formal security analysis for the proposed system. Other future work is related to exploring ways this model can be leveraged in autonomous systems or micro-accreditation (a way to connect students with employers) to aid in assigning rigor to knowledge units in courses. We realize that it is important to establish conditions under which the proposed system will function robustly, along with experiments to confirm the corresponding results.

CHAPTER 5

PROOF OF REVIEW: A NEW CONSENSUS PROTOCOL

5.1. Introduction

We present a novel consensus model called *Proof of Review* (PoRev), which combines concepts from Proof-of-Stake (PoS) and Proof-of-Reputation (PoR). The participants of the proposed blockchain system agree on evaluated reviews, which will be added as transactions to the system. The reviews and the related evaluation data are stored on the blockchain. The reviews drive the reputation model, where reputation is used to regulate a user's participation in the system. PoRev is a tool that can be used to aid applications by ensuring honest and unbiased reviews and assessments, hence providing an immutable and transparent record of data related to both the reviews and the reviewers.

The motivation to introduce such a system is to derive a trust in the participants' reputations through a consensus of their evaluated reviews. In doing that, applications could use this data to apply trustworthy weighted calculations towards an overall value of something or a rating. This system lends itself easily to decentralized marketplaces, where the trust in the reviews and reviewers is important to gauge the objective value of a product or service. Our model could be used to mitigate some common challenges in a review system (bad reviews, spamming, bias, and other errors). In micro-accreditation applications, the rigor of a course could be settled from the aggregation of peer-provided reputation-weighted scores. Currently, comparing courses from one academic institution to another, determining what level of knowledge has been gained, or which school provides a more rigorous application is either done with bias or by humans in an assigned authority to do so. PoRev could be leveraged to in trusting that analysis by trusting the reputation of those doing the work. In sensor-array systems, a consensus on a node's assessment being incongruent could indicate a failure in a sensor. Furthermore, we see how this problem also applies to the future of autonomous vehicles that caravan together in an ad-hoc manner. Artificial Intelligence (AI) is making decisions (routes, re-routing, speed, safety actions, etc.) and other assessments to

deliver a vehicle safely from point A to point B. Currently, autonomous vehicles are making these decisions independently using the data retrieved and the sensors onboard. The trust in that vehicle's AI decisions should not be blind and independent, since any failure in the sensor hardware or in receiving data could be detrimental to its success and possibly dangerous to others. There is a wide range of applications where this system can be applied, outside of the common review (as for products or services), especially with community-based blockchain systems, such as neighborhood watch [**61**], and the others mentioned above.

The proposed system uses the review as a "contribution" which entails a reward for the respective honest user in terms of increasing their reputation. The reputation in turn will function as the user's asset which increases their influence in the system.

The work presented in this chapter is to appear as "Proof of Review - Trust Me, It's Been Reviewed" in proceedings of the 5th ACM International Conference on Blockchain and Internet of Things (BIOTC 2023) [**75**].

5.1.1. Comparison to The Existing Consensus Mechanisms

Let us compare the proposed consensus mechanism to the existing ones.

Proof-of-Work: The major difference is a type of the contribution: review vs. hash power. Our system lends itself naturally towards the Byzantine Agreement like consensus which is used in some Proof of Stake systems such as Algorand [**16**], by using committees to determine a block leader (instead of a hash power based competition). The latter types of consensus models are known to be more cost-efficient and environmentally-friendly compared to PoW systems such as Bitcoin [**50**].

Proof-of-Stake: A natural similarity to these types of systems is seen when directly mapping reputation to stake (as it is done, e.g., in [**60**, **74**]. Then, the components such as committee selection and block mining work in a similar manner as in PoS systems. Some major differences from the currently popular PoS systems is their focus on financial applications rather than community-based applications (such neighborhood watch [**61**] and similar ones such as data cooperatives—see the references in [**61**]). The community-based blockchain systems may be seen as specialized data repositories and this work can be seen as a step towards

tuning blockchain systems in this direction. In particular, in such the systems, it may be easier to detect a misbehaving user hence taking a corrective action upon the committee's approval. In particular, we introduce the blacklisting and minimum-reputation component which results in decreasing of the number of malicious players in the network, and mitigating their actions and extent of their influence on the system.

Proof-of-Reputation: Proof of Review is similar to this type of consensus as there is no mining process and reputation is used in lieu of tokens (money) to determine block forging. PoRev can be seen as an extension of Proof of Reputation, where the reputation system is only one component of the consensus protocol. Compared to PoR system of Gai et al. [**26**], PoRev reputation model and associated values persist beyond a round's block being added. We use the reputation values of the nodes at the beginning of a round, instead of the top-ranked node at the end of the round. Our nodes' reputation values are calculated from each node's history of submitted transactions since initially joining the network, instead of calculated only from the ratings given in each transaction within that round's block. Instead of transactions being ratings (reputation scores) given by other nodes' raters (humans), a review transaction in our system contains the review, the analysis of that review, and the direction on how to adjust the reputation of that reviewer (increase or decrease). Our review evaluation is done automatically with the purpose to exclude a human factor (e.g., a possible bias).

5.2. Related Works

A large body of works on Proof of Reputation and Proof of Stake exists but they do not quite deliver on the automation, flexibility, or trust we need. Let us briefly discuss some of these works which are most closely related to ours.

Bashar et al. [**11**] introduce a process where the role of the block leader is split into multiple parties validating transactions before adding to their own potential block. Then, a union is performed on these blocks to create a master block, signed by each party. This expedites the block-creating process and helps mitigate individuals from omitting or altering a transaction, since all parties involved have to agree on master block. Our focus is on

stopping "bad" transactions by mitigating malicious players from being involved as quickly as possible.

Khan et al. [**48**] introduce a method to identify a transaction malleability attack (where someone changes the hashed ID before the transaction can be validated) in order to study how to protect from it. They do this by adding a secondary layer to the blockchain that maintains transaction-provenance (origin) to check transactions against. Our focus is more general, but could see incorporating this as part of an evaluation process.

Schaub et al. [**62**] propose a new blockchain-based reputation system to preserve privacy in a trustless environment. They focus on seller (a service provider) reputation in e-commerce applications, letting customers to give feedback as both a numerical rating and a textual comment. Reputation is based on an aggregated functionality from all reviews of the service provider. Their protocol does not evaluate the review itself, which is different than Proof of Review (PoRev). Instead, in our system, reviews are evaluated to determine any modification to the reviewer's reputation. Additionally in PoRev, reputation for a user is the summation of all reputation adjustments to the current round for that user.

Kleinrock et al. [**35**] introduced a reputation-fair lottery to be used for the Byzantine Agreement based PoR blockchain with an auxiliary "fallback" Nakamoto based blockchain. Differently from our work, they did not specify the reputation system, and just assume using one which satisfies certain properties.

Larangeira [**37**] introduced a reputation-based trust delegation layer over a PoS blockchain. This layer allows groups of users to assign their trust to arbitrary participants (trustees). This work introduced a concrete reputation system, but differently from our work, the reputation system is functioning at the "application" layer while relying on the PoS ledger at the "blockchain layer".

Salau et al. [**60**] use reputation as a stake in the context of data cooperatives [**29**]. The idea of using reputation to incentivize the parties' honest behavior—which originates from earlier works such as [**26**]—is employed in our construction as well. However, we apply it in a different context of evaluating review credibility. Also, the work [**60**] uses a different

blockchain system, the Snow White protocol.

Leonardos et al. [**38**, **39**] introduce weighted voter profiles in which a round's block is decided by the weighted majority of participants for that round. They leave the stake, committee selection mechanism, and the reward system intact in a proof of stake (POS) blockchain platform to concentrate on their contribution. Their focus is an extension of the "Optimal Weighted Voting Scheme" defined by R. C. Ben-Yashar et al. [**51**], which offers an alternative to typical majority rule (e.g., 2/3 super majority) and allows for a dynamic consensus threshold that adapts to the players selected in a round. A participant's weight is determined as a percentage [0.0, 1.0], calculating the number of times they voted consistent with their peers compared to the number of times they were selected to participate. i.e. The more frequently they agree with the majority of their peers, the higher their weight would be.

In [**38**, **39**], the weight is defined as the probability the player will vote correctly and is adjusted at the end of each round. A wrong decision/vote may be malicious or accidental, such as being temporarily offline, or high latency. In their setup, any node with a weight less than 0.5 is suspended from being selected for the round. They show this scheme, through example, improves consensus times by scaling the vote threshold dynamically and mitigating scenarios where it is not possible to reach a 2/3 majority due to the number of low-weight participants. Though solving one problem, this introduces new vulnerabilities such as maintaining anonymity, and issues of recovering from suspension. Although this paper introduces an approach of weighted votes to essentially and gracefully "blacklist" participants, it only focuses on the consistency of consensus protocol votes compared to the majority in that round. On the contrary, our research blacklists users for both technical (e.g. bad signatures, double spending) and grievous behavioral infractions like spamming, ganging-up, bias. Additionally, our blacklisting is not directly related to the voter's reputation (or weighted voter profile in this paper).

Pal et al. [**54**] introduces a decentralized solution to a centralized ride-sharing platform that helps ensure a fairness in the evaluation of a ride(complaints and reputations of all

participants) as well as a mechanism to negotiate fare payments. The paper discusses their blockchain-based system (BlockV) which addresses both smart-contracted paths/fare costs and payments between the driver and rider, and a complaint action that affects both reputation and monetarily. They use a built-in reputation system where driver gains reputation through a successful completion of a ride. The driver loses reputation if a complaint is made and found that they had deviated from the agreed contracted route. The rider is penalized monetarily if they enter an unjustified complaint. Validating road side units (RSU) are used as an external oracle to record locations of participants during the route, and is used for comparisons against the decentralized route fare database (RFD) which contains all the possible routes and fares for computation and determination. Complaints regarding behavior, or any other type outside the technical contracted route, is not addressed. Location and reputation is used by the rider to select a driver to contract with, though this is still a manual choice and completely decided by the rider. Alhough this paper shows a novel way to maintain transparency in the ride details, including routes, fares, and complaints, it only focuses on the technically malicious actions and consequences of the parties. On the contrary, we also want to focus on the behavioral maliciousness and analysis of the reviews left for both the driver and the rider.

Li et al. [**41**] propose a decentralized system to mitigate the effects of compromised routers. Their paper suggests using a blockchain to manage, calculate, and store reputations of routers, using the immutable data to evaluate these reputations based on their history of behaviour (the trustworthiness of the router). Their goal is to remove the single-point failure of currently centralized Reputation Management (RM) solutions to improve IoT systems' QoS and without needing complete trust in any external third party. This blockchain system (BC, and referenced as the Edge Server Layer) exists outside of the router network layer. Neighbors of the data-forwarding router (DFR) submit reports of that DFR to the nodes on the blockchain as transactions, and contain the information needed to determine whether it's acting cooperative or not. This is called the cooperative feedback and is used in the calculation of a router's reputation in the next BC round. The routers' reputation values

are stored on the blockchain. If a reputation falls too low, the router is blacklisted, with the BC nodes broadcasting to the routers to avoid this router. They showed that their approach significantly reduces the impact of malicious (compromised) routers within an acceptable number of transactions, when up to 50% are malicious. Additionally, they show that the average reputation value increases quickly as the corrupted routers are blacklisted over 1000 transactions.

5.2.1. Overview of Our Contribution

We propose a Proof of Review system which integrates the reputation system into Algorand's PoS engine, and we add the two useful components which enable blacklisting and enforcement of minimum-reputation requirements.[1] Specifically, we update and modify the Algorand core to implement our new protocol. For security analysis, we argue that our modification to the Algorand core preserve the properties of the original PoS system up to adjustment of some parameters, so that the resulting blockchain system remains secure under an assumption that up to 1/3 of total reputation (instead of stake in the original Algorand) is controlled by the malicious parties. We also release our code and show the simulation results which confirm that liveliness is assured and consensus times are not significantly affected by requiring a minimum stake to participate. We also confirm blacklisting does not affect liveliness or completeness (that a transaction is guaranteed to be added to the blockchain eventually) when $< 75\%$ of nodes are blacklisted. In these tests, for our review validation and reputation adjustments, we use the same NLP component as in [**74**]. However, we note that in practice, any other application-specific components can be used.

In this paper, we depart from the Algorand construction because we are changing the engine to construct a system that is more suitable for our purposes, which is review systems.

Instead of using Algorand in the off-the-shelf model—as it is done in [**74**]—we modify the engine by generalizing the validators (we use NLP, but other evaluation systems can be used for other purposes), adding Proof of Review, blacklisting, and re-assembly of

[1]We note that the minimum stake handling in Algorand does not quite satisfy our goals. See Section 5.3.5 for discussion.

transactions in a block before being it is added to the blockchain. Since, we assume that the participants belong to a community of users contributing towards a certain goal, the assumption that no more than 1/3 of the total reputation is controlled by the dishonest parties seems to be reasonable.

In our system, a correct review serves as a participant's asset (we will call it a "contribution"). Bitcoin's contribution is hash power. PoS contribution is a token. In PoRev, contribution is doing the "right thing" that earns reputation. In Proof of Review, as a special case, the "right thing" is providing a congruent review, meaning that the review has an adequate evaluation as compared to the ground truth value.

5.3. Our Proposal

First, we discuss our proposed architecture, which generally follows the construction of Algorand, focusing on their differences. Then, we describe in detail the new components, which allow the proposed system to function effectively in the context of review systems.

5.3.1. Overview of the Proposed Architecture

In our proposed system, we modified the consensus protocol model to implement Proof of Review, with components and mechanisms such as working with reputation (instead of tokens) and a evaluation mechanism (we chose to use an NLP system like [**74**] does, though other types of evaluations can be done for other types of reviews).

The reviewer's reputation is adjusted up or down depending on that evaluation at the end of the round, and the ability to participate further in the system (beyond writing reviews) is dependent on that reputation. In this context, participation in the blockchain system is the ability to create or validate blocks to be added to the blockchain. These blocks contain the reviews, the analysis of the reviews (added at the block proposal stage), as well as other associated data. The users with the most consistent reviews are more likely to have earned higher reputations and therefore chosen more frequently to participate. In contrast, if the parties produce too many incongruent reviews, the reviewer's reputation will significantly decrease, and they may be restricted from participating in the consensus process

due not meeting a minimum reputation stake. This will be discussed in more detail in the next section.

First, a user joins the network by writing and submitting a two-part review, a text comment and numerical rating. An account is created with generated public and private key pairs, and an initial reputation (value of 1) is given and associated to their public key. When a review is submitted, the system creates a review transaction and it is broadcast to all the nodes' transaction pools awaiting further action.

Next, a potential block leader gathers these transactions, and submits each review to an evaluation system for analysis (Fig. 5.1, Step 1). In our work, that review is evaluated to determine if it is a "congruent" one, that is that the review text is consistent with the review rating. Evaluation also looks for malicious actions such as spamming, trolling, or acting in a biased manner (the properties that make up a congruent review are discussed later). Once the evaluation is complete, those results and reputation adjustment info are added to the transaction, and it is added to the leader's potential block. At the end of Step 1, all the potential blocks are broadcast, and in the next step, a block leader is selected, vote on, by a new committee in the same manner Algorand does.

In the next step, the block is verified. Meaning, each review transaction is opened and the review is re-evaluated using the same evaluation mechanisms the leader used. This is performed to ensure agreement with the leader. If there are inconsistencies in the evaluation of any reviews between the verifiers and the leader, the verifier will consider the leader as malicious and vote to blacklist the leader (using a flag in their next message). In addition to that flag, they will vote for an empty block, since that leader's block of transactions is considered "bad" now.

At step 4, If enough messages are heard from the previous step is to blacklist, then that message (the flag and the vote for an empty block) will be propagated onward to the final step. The other mechanism employed during this step are handled the same way as Algorand. In the final step, if the block is not empty, then each review transaction is applied by adjusting the reputation of the reviewer. Any blacklisting flags will also be applied.

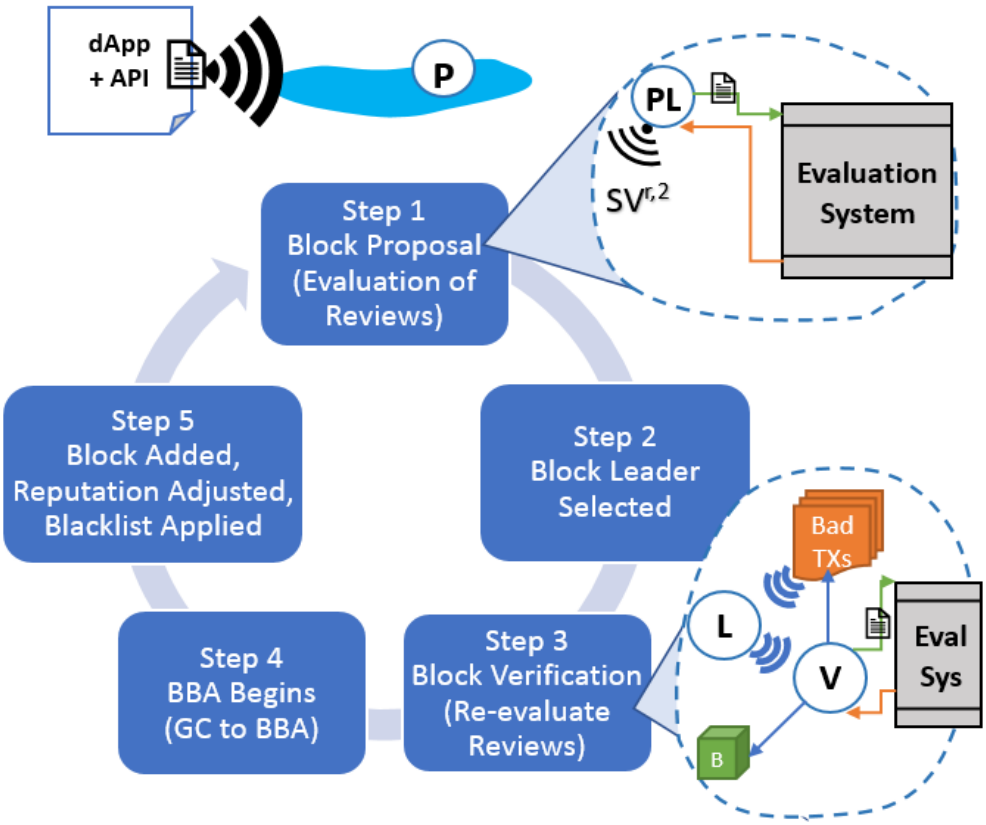

FIGURE 5.1. Steps of Proof of Review.

At the end of the round, the block is added to the blockchain, with the new state which reflect the new reputation values.

The data forged in the blocks can be used by various dApps for further handling at a higher level, if desired. Examples of how dApps may use this data include showing the review along with its evaluation, automatically hiding low-reputation reviewers from the public until manually approved, calculating an overall item score from reputation-weighted reviews, and many others. Proof of Review is a dynamic, self-regulating, general-purpose tool for various types of reviewer systems. Similarly to other consensus models, it is used to help prevent malicious parties from dominating the network in addition to providing supplementary information for dApps to use if needed.

5.3.2. Details of the Proposed Architecture

We select Algorand as a basis of our design, because it is open-source and it was easy to adapt its Proof-of-Stake mechanisms such as the committee selection, stake calculation, and other mechanisms for our purposes. The modifications which we made to the Algorand system are described below.

Maintaining Reputation. Reputation is used in calculating the stake, instead of tokens (money). Reputation, denoted as $\mathcal{R}$, is mapped to stake similar to how it is done in [**60**, **74**]. Reputation[2] is associated with each public key pk, which is generated and owned by each user. The initial reputation value given a new account is 1 (an integer). The stake is calculated based on the reputation of a user in the system proportional to the total reputation in the system at the beginning of a specified round r. Unlike Algorand, reputation (and hence the stake) cannot be increased through purchase or trade. Reputation must be earned (mechanism explained later). This stake is used as one of the inputs for VRF Sortition for committee selection and also in calculating weighted votes during consensus.

Let us define the weighted proportional stake as follows:

$$\omega_i^r = \mathcal{R}_i^r / \mathcal{R}_{PK^r}^r,$$

where $\mathcal{R}_i^r$ is reputation of a user i, PK^r represents the public keys of all active users at round r, and hence $\mathcal{R}_{PK^r}^r$ is the total sum of reputation of all active users at round r. Simply, a user's weight is calculated from round r reputation for that user i divided by the total reputation in the system of all active users. Offline and blacklisted users' reputation is not considered.

Public Ledger. In our system, it is extended as follows. Making payments is now an ancillary feature instead of a primary one. This evolution fundamentally changes how we present this attempt at an idealized public ledger and its **Initial State**. Users may join the system whenever they wish by generating their own public and private key pairs by submitting a review. Upon joining, an account is given an initial reputation value of 1.

[2]In principle, tokens can also be associated with a public key but is not necessary for our system to function.

Let $pk_1, \ldots, pk_j$ be the set of initial public keys, $\mathcal{R}_1, \ldots, \mathcal{R}_j$ be the set of initial respective reputations, $md_1, \ldots, md_j$ be the set of initial respective private metadata. In other worlds, at the start of the network, the sets of public keys, associated initial reputation value, and associated metadata are publicly known. With this, the modified initial state can be simplified to

$$S_0 = (pk_1, \mathcal{R}_1, md_1), \ldots, (pk_j, \mathcal{R}_j, md_j)$$

We remark that the Algorand tokens are still present as a part of transaction, but they are not used by our system, and hence we omit mentioning them.

Review Transaction ($ReviewTX$). It is derived from PaymentTX as defined in [**15**], but in our system, it does not require payment. This is similar to the review transaction in [**74**], but we codify the fields into newly added public and private metadata (discussed later). A pk (reviewer), for a review transaction $\mathcal{REV}$ where I represents non-sensitive information (e.g. review fields) and the $\mathcal{I}$ is additional information that is considered sensitive (e.g. identifiers, private metadata defined later in this section, etc) and hashed to protect it. The review fields may be empty and not used in every transaction, except for two fields (ReviewNote and ReviewRate) which are mandatory to submission.

$$\mathcal{REV} = SIG_{pk}(pk, I, H(\mathcal{I}))$$

Potential Leaders (PL) actions now evaluate the review within a Review Transaction before adding it to their block through the help of an evaluation system. In our instance, we employ NLP as our evaluation component like [**74**] does. With this, a review is evaluated for congruency between the review text and the review rating (checking for fake reviews). With other evaluation components, a review may be analyzed for correctness or consistency, like in a sensor-array network, the data could be checked for validity against other sensors or in another manner.

A Verifier Committee (SV) later evaluates the same reviews in the Leader's block, then compare their results to the Leader's results. This is in addition to the common technical validations (e.g. digital signatures, etc.). If the block is valid, and the Leader's evaluations

are consistent with a Verifier's, the Verifier will vote in favor of adding the block to the blockchain; otherwise, not in favor. If the majority of the committee votes in favor, then then the block will be added at the end of the round.

Minimum Reputation is required to be selected for a committee (note that reputation is mapped to stake). Every round and every step maintain the following: Let ω_i^r represent the proportional stake of user i for Round r. Every honest verifier

$$i \in HSV^{r,s} \wedge i \in \{PK^r : \omega_i^r \geq \omega_{min}^r\}$$

where ω_{min}^r is chosen in a way to prevent malicious new users access to participate, while still ensuring that $SV^{r,s} \neq \varnothing$.

A Reputation Adjustment algorithm uses the evaluated congruency (mentioned above) to calculate whether the reviewer's reputation should be adjusted positively or negatively. Finer details of the mechanism are explained later. Additionally, at the end of the round before a block is added, another mechanism adjusts the reputation of each reviewer. The reputation state of all users is stored in the ledger. These adjustments consequently could affect a user's stake. We employ the additive increase/multiplicative decrease functionality in our algorithm similar to how [**74**] does.

An analysis for behavioral maliciousness is added to the existing technical checks to mitigate corrupt dishonest nodes further. This is defined in a later section "Technical and Behavioral Honesty Component".

New metadata (*md*) is added to an account and associated with each public key *pk*. The *md* is divided into two types. The private type fields include originating institution, organization, country of origin, group membership, etc. and is used in detecting certain malicious behaviors (e.g. bias). This data is added to a ReviewTX and becomes associated with the review and reviewer. The public fields include the account's number of reviews, timestamp of the last and 100th prior review, blacklisted information (described below), etc. and used in detecting other malicious behaviors (e.g. spamming, bot activity).

Blacklisting mechanism is employed for most malicious behaviors (defined in a later subsection) instead of simply ignoring nodes. This is voted on by a committee (a new "flag" is propagated with a message to the next committee indicating which account). The blacklisting information is part of the public *md* and includes the following fields: BL (Boolean flag), the number of rounds BL lasts, Timestamp of most recent BL, and the number of times account has been BL. New methods are added to the technical checks on committee membership to check if an account is blacklisted. An SV will ignore messages from determined blacklisted accounts.

5.3.3. Proof of Review Consensus

Steps of Proof of Review: (1) Evaluation of reviews; (2) Selection of the round's leader and their associated block containing reviews; (3) Reevaluation of reviews; (4) Block of reviews agreement (block decision); (5) Block addition to the ledger, reputation adjustment, blacklisting (see Fig. 5.1).

A round starts regardless of any transactions in any transaction pool and ends in a new block (even an empty one) being added to the ledger. Transactions are created and propagated to the network using a dApp (web-based, console-based, mobile-based, desktop-based). A reviewer on the network (or newly joins the network, if not) uses this dApp to submit a review containing both text and a numerical rating. The transaction is created, signed, and disseminated to the network.

Step 1: Block Proposal. Potential Leaders gather review transactions (vs payment), verify their technical details, use an evaluation system to evaluate the review within the transaction TX, add the evaluated score and reputation adjustment to the TX, then add them to their payset of their proposed block. If an evaluation shows malicious behavior (like spamming), a blacklisting flag within the transaction will be set true for that reviewer. Blacklisting reviewers is applied at the end of the round.

Step 2: Block Leader Selection. Methods and communication to and from Step 2 Verifiers remain the same as Algorand.

Step 3: Block Verification. The block is certified. A Verifier Committee waits for a certain number of messages for v (a vote for the leader). Once receiving enough votes, the Verifiers check the technical details of the block, evaluates the review transactions in the Leader's block, then compares their results to the Leader's results. If the block is technically valid, with the Verifier's and Leader's evaluations also being consistent (including any flag for blacklisting reviewers), they broadcast a vote for that Leader and the proposed block. If the block is not valid, it's same as Algorand (with one exception).

If any review transaction evaluation is inconsistent with the Leader's evaluation, the Verifiers vote to blacklist the leader and submit their vote for an empty block.

Step 4: Binary Byzantine Agreement. Methods and communication to and from Step 4 Verifiers remain the same as Algorand.

Step 5: Block Decision. If a block decision was in favor of the Leader's block, application of review transactions' reputation adjustments are performed. Any reviews flagged for blacklisting are applied to the associated reviewer. Otherwise, the block is an empty block, and if enough votes are heard to blacklist the Leader, blacklisting is applied. Finally, the block and reputation state are added to the blockchain.

5.3.4. Integrating the Blacklisting Component

We extend the definitions of an honest and malicious user to include *behavioral* honesty in addition to technical honesty. Most players will be technically honest since a lot of research has been done to catch or prevent this from occurring in a system. We deem a player to be acting behaviorally dishonest when there exists a pattern of bias, bullying, ganging-up, stuffing the ballot, and more (this list is not exhaustive). The technical honesty definition and the prevention of maliciousness are addressed by Algorand and remain valid when it is extended by our reputation model. Furthermore, in extending the definition of Honest and Malicious to include both technical and behavioral models, this does not weaken the security of the Algorand's original technical model. The same requirements and conditions still hold for correctness, completeness, and soundness. The addition of a reputation model does not

impact the sets of honest or malicious verifiers beyond further constraints such as not being blacklisted. Instead of merely being ignored, both technical and behavioral maliciousness (e.g. bias, ganging-up) is handled punitively by blacklisting the participation of the user responsible for a certain number of rounds. This is in addition to reviewers' reputation adjustments which encourage reviewers' consistent reviews and discourages incongruous ones through a negative reputation adjustment.

A user is considered to act behaviorally honest when they submit a review where:

- The review text is consistent with the review rating, and
- The review text is unbiased and honest, and
- This review and the user's immediately previous review is consistent with an expected human time interval, and
- The user is acting independently and for themselves in submitting this review.

A user is considered to act behaviorally malicious when they:

- Submit a review, where the review text is incongruent with the review rating (e.g. 1 star, "Great book!"), or
- Act in a biased way, like always giving positive (or negative) reviews for specific people, types of people, items, ideas, etc. regardless of the quality of what they are reviewing, or
- Submit a review inconsistently irrelevant with what they are reviewing, acting like a troll or bully, or
- Submit multiple reviews in succession within a time interval either incompatible with a human ability (e.g. using a bot), or in such a way that reviews are likely of discordant and inadequate value (e.g spamming), or
- Submit a review incongruent with the majority of other reviews. Being an outlier cannot absolutely establish that a user is acting maliciously since the user could be acting honestly and independently. Other factors, like the frequency of this behavior over a certain time interval, may indicate maliciousness.

We now address how the different types of behavioral maliciousness is mitigated

and handled. While incongruent or dishonest reviews lead to a reduction in reputation, technical and behavioral maliciousness is handled by blacklisting a node. Reasons to blacklist include bias, bullying, ganging-up, stuffing the ballot, and more (this list is not exhaustive). Blacklisting prevents a node from self-selecting for any committee and can be verified as being blacklisted by other nodes. Nodes are blacklisted by majority vote for a certain number of rounds. The blacklisting information is part of the public metadata associated with an account (defined in an earlier section). New methods were added to both blacklist an account and to verify blacklisting status.

5.3.5. Integrating the Minimum Reputation Component

Our minimum reputation component aims at cutting under-performing users from gaining influence in the system. In our implementation, each new user is assigned a starting reputation value of 1, and then this value is being increased if the user makes their contributions to the system in the form of congruent reviews. Such the users will eventually gain enough reputation to have a chance to be selected as block leaders. At the same time, the users who are idling or who lose reputation, e.g., due to incongruent reviews or due to incorrect behavior described in the previous section, will eventually lose this opportunity.

Note that in Algorand, the minimum stake handling does not directly addresses the problem. It requires an account (a node) to own greater than zero tokens to potentially participate in the network; otherwise, the account is closed. It cannot be directly translated into our reputation scenario our system, because our community-based scenario allows for zero-reputation case. Even though, a node may have reputation below the minimum, its account is not closed and it still has a potential of playing in future rounds.

In our implementation, we use a simple ad-hoc approach by taking the minimum reputation value to be the median of reputation values of all users in each round. This way, we allow a substantial fraction of nodes to participate assuming that the reputation distribution is reasonably even. That is, the distribution of potential leaders will be skewed if there are a few nodes with very high reputation. In this case, one may apply the reputation-fair lottery of Kleinrock et al. [**35**] to resolve the issue.

5.4. Security Analysis

Algorand's security proof can be adapted to our proposed architecture. Public and private keys are generated the same way as Algorand, and verifiers use the same mechanism to sign messages. Unlike Algorand, we have limited participation to users with a minimum stake/reputation who have not been blacklisted. The minimum stake is factored into the committee selection and checking if a user is blacklisted applies to both self-selection and credential verification when receiving messages from previous steps' nodes. As we will establish in the next section, adding the blacklisting and minimum-reputation components does not significantly affect the timing of the underlying blockchain system. In this section, we argue that our modifications effectively preserve the security properties claimed in [**16**].

For the adversary model, similarly to Algorand, the adversary can corrupt any node, control the corrupted nodes' actions, and can create new nodes that join the network at any time. We assume that the probabilistic polynomial-time bounded adversary is able to control strictly less than 1/3 of the total reputation in the network. Specifically, in any round, the adversary can only control the set of parties such that the sum of their reputation values $\mathcal{R}_i$ is less than $\mathcal{R}_{TOTAL}/3$. Though an adversary can make the nodes of the system wait to receive messages the maximum amount of time allowed, they are still bound by those time limits indirectly. Honest nodes will simply stop waiting after the system determined time (as discussed in the Steps earlier, and below in definitions of λ and Λ). Additionally, the adversary can submit reviews as frequently as they can, and they can coordinate in submitting multiple reviews simultaneously at any time. It should be noted that the adversary is not assumed to be aware of the real-world identities of any participant other than themselves.

Since we are altering and extending Algorand's engine for our protocol, we examine Algorand's Theorem 1 and associated security properties from [**16**]. We borrow the following notation from their work:

- λ and Λ are the time bounds for sending one message and upper bound time to propagate a non-empty block (i.e., the maximum time that the block reaches the

majority of nodes), respectively.

- T^r: the time when the first honest user is sure about B^{r-1}, the previous round's agreed upon block.
- I^r: the time interval $[T^r, T^r + \lambda]$.
- $\alpha_i^{r,1}$ is the time a user i starts step 1 of the round. $\beta_i^{r,s}$ is the time user i ends step s of the round.
- h is the fraction of honest users in the system (2/3, 1].
- $\mathcal{P}_h$ is the probability of the leader ℓ^r is honest $(0, 1)$.
- PAY^r is the payset within the block for each round r. In our proposal, the leader gathers review transactions and adds them to the block's payset.

The following theorem is adapted from Theorem 1 of [**16**].

THEOREM 5.1. The following properties hold with overwhelming probability for each round $r \geq 0$:

(1) All honest users agree on the same block B^r, and all transactions in B^r are valid.

(2) When the leader ℓ^r is honest, we have the following.
   - The block B^r is generated by ℓ^r and all honest users know B^r in the time interval I^{r+1}.
   - If $PAY^r = \emptyset$ then $T^{r+1} \leq T^r + 6\lambda$; otherwise B^r contains a maximal payset received by ℓ^r by time $\alpha_{\ell^r}^{r,1}$ and $T^{r+1} \leq T^r + 4\lambda + \Lambda$.
   - Let $B^{r'}$ be the last block before B^r with a non-empty payset. If $\ell^{r'}$ was honest or if $T^r - T^{r'+1} \geq \Lambda$, then $PAY^r \neq \emptyset$.

(3) When the leader ℓ^r is malicious, all honest users become sure about B^r in the time interval I^{r+1}, and $T^{r+1} \leq T^r + (6L + 8)\lambda + \Lambda$.

(4) $\mathcal{P}_h = h^2(1 + h - h^2)$ for L^r, and the leader ℓ^r is honest with probability at least $\mathcal{P}_h$.

We would like to argue that the changes, which we introduced to Algorand, did not substantially affect the protocol flow. Therefore, we are able to use a very similar security

proof to argue security of our proposal.

To summarize, the main changes, which we introduced to Algorand are review transactions, an evaluation and reputation system, and in addition, we introduced the blacklisting and minimum-reputation components.

Although properties 1 and 2 remain unchanged from Algorand, we did change the time-bounded variables and increased the values for λ and Λ (in Algorand's parameter system file) to offset the extra time needed to do evaluations on the reviews. Without those changes, while running our simulated tests, the system "timed out" and continually added empty blocks ($PAY^r = \emptyset$) to the blockchain. Since only values of the time-bound parameters changed and not what they represent theoretically, Properties 1 and 2 in our theorem hold similarly to [**16**].

At this point, we can borrow the Completeness Lemma (Lemma 5.3 of [**16**]), which is stated as follows.

LEMMA 5.2 (Completeness Lemma). Assume Properties 1–3 of Theorem 5.1 hold for rounds $0, ..., r-1$. When the leader ℓ^r is honest, Properties 1 and 2 of Theorem 5.1 hold for round r.

Next, we note that Properties 3 and 4 are affected by introduction of the blacklisting component. Specifically, in Step 3 (see Section 5.3.3), when the potential leader ℓ^r is considered technically or behaviorally malicious, the set of verifiers vote to blacklist ℓ^r, which removes the leader from further affecting the consensus protocol. (We discussed these blacklist-worthy behaviors in detail in the previous section.) Let us now discuss changes to the above mentioned properties below. We will argue that the Soundness Lemma of [**16**] will still hold taking into account those changes.

LEMMA 5.3 (Soundness Lemma). Assume that Properties 1–3 hold for rounds $0, ..., r-1$. When the leader ℓ^r is malicious, Properties 1 and 3 hold for round r.

PROOF. Let us analyse how blacklisting affects the probability of an honest leader being selected.

In [**16**], the authors prove the Soundness Lemma (when ℓ^r is malicious) by analyzing the SV messages during the graded consensus protocol GC and BBA* at Step 4+.

Differently from their construction, our protocol introduces blacklisting of ℓ^r when they are malicious. In Step 3, $SV^{r,3}$ will vote for an empty block and their message will include a flag to blacklist the leader. In Step 4, if enough votes are sent and agreed on for blacklisting ℓ^r, the protocol follows Property 2 maximum time definition when $PAY^r = \emptyset$, because the leader and their potential block are then ignored. The participants continue by voting for an empty block (with an empty payset) in their messages.

$$\text{If } PAY^r = \emptyset \text{ then } T^{r+1} \leq T^r + 6\lambda$$

We introduce our new Property 3 as stated in Theorem 5.1: when the leader ℓ^r is malicious, all honest users become sure about B^r in the time interval I^{r+1}, and

$$T^{r+1} \leq T^r + 7\lambda < T^r + (6L + 8)\lambda + \Lambda.$$

By the proof of [Lemma 5.4] [**16**] proof, a malicious leader can force honest users to wait the maximum amount of time 2λ at each step after Step 2, and an honest leader and verifiers send messages more promptly within time at most λ.

In our system, a malicious leader ℓ^r only affects the maximum time for Step 3 due to the way the blacklisting works (e.g., vote for empty block and payset $PAY^r = \emptyset$). For sake of argument and our proof, we assume the majority of verifiers to be honest if the majority vote to blacklist the leader. Since honest verifiers send messages as prescribed in the protocol (that is without delays), ℓ^r being malicious or honest is irrelevant in regards to the maximum time for each remaining step **after** Step 3. Therefore, 2λ is the maximum time that Step 3 can take (since the malicious leader might be forcing the maximum time). Hence, λ is the maximum time for Steps 4 and 5: recall that blacklisting causes a change in vote to an empty block and payset, so the leader is ignored and so they are unable to affect the maximum time.

According to the above discussion, we set the maximum time for a user to end Step 3 to

$$\beta^{r,3} \triangleq \beta^{r,2} + 2\lambda.$$

When the original $PAY^r \neq \emptyset$ for the malicious ℓ^r, then following the calculation in [**16**], we have:

$$\beta^{r,2} \triangleq T^r + 3\lambda.$$

When we trace the running time from the beginning of the round until the end of step 5 of the round, from definitions above, we now have

$$\beta^{r,3} \triangleq (T^r + 3\lambda) + 2\lambda.$$

$$\beta^{r,5} \triangleq ((T^r + 3\lambda) + 2\lambda) + \lambda + \lambda = T^r + 7\lambda.$$

With all honest nodes seeing this empty block B_ϵ (voted on from Step 3 onward) by $\beta^{r,5}$, within the I^{r+1} time interval, with $T^{r+1} \leq T^r + 7\lambda < T^r + (6L^+8)\lambda + \Lambda$.

In summary, Properties 1 and 3 hold for round r when the leader ℓ^r is malicious, so the soundness lemma holds as well.

□

It remains to argue that the block leader is honest with substantially high probability. We can directly apply the Lemma 5.4 from [**16**].

LEMMA 5.4 ([**16**]). *Assuming that Properties 1-3 of Theorem 5.1 hold for rounds* $0, ..., r-1$, *we have* $\mathcal{P}_h = h^2(1 + h - h^2)$ *for* L^r, *and the leader* ℓ^r *is honest with probability at least* $\mathcal{P}_h$.

The above lemma directly applies to our setting. We would only like to mention that blacklisting malicious users in our system only increases the fraction of honest users h.

Combining the completeness and soundness lemmas as well as Lemma 5.4, Theorem 5.1 follows by induction (similarly to [**16**]).

LEMMA 5.5 (Block Correctness from Honest Leader Lemma). Restricting the protocol participation to users having a minimum stake and not blacklisted increases the probability of an honest leader being selected for each round r.

PROOF. Note from our integration described previously, we map reputation to stake.

Since $\mathcal{P}_h \in [0, 1]$: a constant greater than or equal to 2/3 representing the fraction of reputation "owned"/associated by honest participants, then $\mathcal{P}_m \in [0, 1]$: a constant less than 1/3 representing the fraction of reputation "owned" by malicious participants. The probabilities that the leader selected is honest or malicious are at least $\mathcal{P}_h$ and at most $\mathcal{P}_m$, respectively.

To prove the minimum stake requirement, we consider the following instance. Recall that we use the median of reputation values of all users in each round (M) to determine the minimum stake (ω^r_{min}).

$M \geq \mathcal{R}_{LEAST}$, meaning the more than half the nodes have a reputation greater than the node with the least reputation. With that, $\omega^r_{min} = M/\mathcal{R}_{TOTAL}$, where $\mathcal{R}_{TOTAL}$ is the sum of all reputation of active users in the system for round r

For simplicity in our proof, let's consider an example having 100 nodes and $\mathcal{R}_{TOTAL} = 1000$, with $M = 10$ and the first 49 nodes having a reputation of 5. We also restrict the system to no transactions created and consequently no reputation is adjusted (reputation remains constant). Additionally, we give the adversary more power by allowing participation immediately (versus needing to wait k number of rounds). Then, $\omega^r_{min} = 10/1000 \equiv 1\%$. The first half of the nodes will not be able to participate in the consensus for round r since their calculated stake is $\omega^r_i = 5/1000 \equiv 0.5\%$.

An adversary (A) can create new accounts in an attempt to increase their odds of being selected. Before a new round begins, 50 new nodes appear in the network, all controlled by A. Since every player started at a reputation of one, and the M is 10, these new nodes cannot participate in being a leader ℓ^r (or in any of the consensus protocol) until their

reputation is increased to at least M.

At the beginning of the new round, we have 150 nodes ($\mathcal{R}_{TOTAL} = 1050$, 50 new nodes each with a reputation of 1). In our instance, we can deduce that the M is now 5, since the original first nodes get "bumped" up in sorted location in the reputation set. The new nodes of A do not meet the minimum stake and are prevented from participating, since their reputation is 1.

There may be other instances where A floods the network with enough new nodes/accounts that it allows the new ones to be selected (e.g. 500 new accounts). A is still limited by a low weighted influence in the system. We have shown at least one instance where requiring a minimum stake prevents participation, which proves it increases the probability of an honest leader.

We have previously shown the positive effects of blacklisting users on the probability of an honest leader above 5.4. With this, the Block Correctness from Honest Leader Lemma holds. □

5.4.1. Further Examination of Blacklisting

We introduced blacklisting malicious users in our system, which removes them from the pool of potential participants. We use deduction to show the probability of selecting an honest leader is the same or greater P_h with respect to the blacklisting changes made to the engine.

It can be deduced that by removing these players, it increases the fractional value of honest users to adversarial ones in the system. To show this more formally, let b be the fraction of users blacklisted, h and m be the fraction of honest and malicious users, respectively.

$h : m$ is the fractional ratio of honest to malicious users.

$$h + m = 100\% \therefore hX + mX = 100\%$$

So, without removing any blacklisted b users,

$$(h+m)X = 100 \Rightarrow X = 100 \div (h+m) \therefore X = 1$$

When b is removed from m,

$$hX + (m-b)X = 100 \Rightarrow (h+m-b)X = 100$$

$$X_\Omega = 100 \div (h+m-b)$$

Since $(h+m-b)$ will always be < 100 for any $b > 0$, then

$$X_\Omega = 100 \div (h+m-b) > 1 \therefore X_\Omega > X$$

If $X_\Omega > X$, then $h_\Omega > h$, which proves formally the fraction of honest users h increases when a fraction of malicious users b are blacklisted and removed.

In removing malicious players from the pool, we increase the h fraction which decreases $(1-h)$. In doing this, we deduce that $\mathcal{P}_h$ will increase, and in turn the probability of an honest leader being selected also increases. Though the value of h changes, blacklisting does not alter the probability calculations provided by [**16**] based on location of the first honest user in a random permutation of participants in a round.

As an example, consider a pool of 1500 potential participants, 80% honest and 20% malicious. During the course of one round, 10% (or 150) malicious players are blacklisted and removed from the pool. We apply the numbers to our calculation as

$$X_\Omega = 100 \div (80 + 20 - 10) = 100 \div 90 \therefore X_\Omega = 1.\overline{11}$$

$h_\Omega = 80 \cdot 1.\overline{11} \approx 88.89\%$. This leaves the system with 89% honest participants afterwards.

It is easy to deduce that the probability of an honest leader increases in connection with the increase in b and consequently h. Therefore, if a number of parties are blacklisted, then P_h is better than when they are not.

5.4.2. Analysis of Practical Attacks

We will discuss several types of attacks that are common for Proof of Stake and Reputation systems, and how our Proof of Review model addresses those attacks.

Spamming: In this type of attacks, the reviewer floods the system with reviews that are positive or negative to skew results. These are orchestrated actions which may be set in motion by parties who are wanting to influence a reputation quickly and deliberately, or just wanting to be troll or nuisance. In our model, spamming is mitigated due to an evaluation checks a review's current timestamp against the timestamps of the immediate prior and the 100th prior review (from stored public metadata discussed in Section 5.3.2). The frequency of reviews may indicate spam or even bot activity.

Sybil Attacks: Recall that any adversary entering the system newly is given the starting reputation value. Recall that value is only increased when reviewing is properly done and it remains unchanged when simply being present (idling) in the system. Our model implements a minimum reputation requirement, so that a newly joined sybil node (controlled by the adversary) would not be allowed to participate without putting in some honest work. Hence this attack is mitigated.

On-Off Attack: is where malicious actions may appear to be a random anomaly to avoid detection. Meaning, an adversary could try pretending to be an honest reviewer, increasing their own reputation by submitting many honest, uniform, and unbiased reviews to reach a high reputation. Then, after reaching that high reputation, act maliciously (spamming, corrupt block leader actions, dishonest reviews, etc.). This is mitigated already by our model due to account parameters (metadata), multiplicative-decrease of reputation for incongruent reviews, and blacklisting ability.

Ballot Stuffing / Badmouthing: In these attacks, the reviewer inundates the system with reviews in an attempt to influence the perceived value of what is being reviewed. The purpose of the adversary in a ballot-stuffing attack is to lift an entity's reputation to fool others into a false sense of trustworthiness. Bad-mouthing is the opposite, where parties are

attempting to undermine the trustworthiness by quickly flooding the system with negative reviews. Our model mitigates this through an evaluation of the review plus using the same mechanisms discussed to prevent spamming.

Bias (i.e. always giving great or poor reviews for a brand, person, or institution origin) is handled through blacklisting. The evaluation of a review is checked against the history of the reviewer using the metadata of what is being reviewed (public metadata discussed in Section 5.3.2). Since submitting a seemingly biased review cannot absolutely establish that a user is acting maliciously, other factors, like the frequency of this behavior over a certain time interval, may indicate maliciousness. This was not explicitly implemented, but theoretically we add here for discussion.

5.5. Simulation Results

5.5.1. Testbed

We implemented the proposed Proof of Review consensus by modifying the Algorand blockchain system as explained above. This allowed us to validate our system as well as to demonstrate the performance of blacklisting and minimum-reputation components.

Specifically, the engine of Algorand and its associated SDK code was modified as described in Section 5.3, and the resulting is made available in [**2**]. A new metadata struct was associated with each node (see section 5.3.2 for more detail), including blacklisting data such as whether a node is blacklisted, for how long, and how many times it was blacklisted in the past (described further in section 5.3.2 "Blacklisting"). In particular, we added extra queries and conditional statements in the committee selection process that checked if a node is blacklisted and whether the node's reputation meets a required minimum. Since Algorand already had functions to retrieve total tokens of active users in the system (and specific user's token quantity) for its calculation for a user's weighted stake, we copied and modified them to retrieve reputation values instead. We used our functions in place of theirs, and employed our calculated reputation-based weighted proportional stake as one of the inputs for committee selection and determining whether the user meets minimum stake.

We needed a way to test our new functionality in an easy and controllable way. We decided on the distributed app (dApp) used from [**74**], since it had a re-configurable graphical interface and its simplicity to reprogram for testing parts of our protocol. Additionally, this dApp already had a mechanism to output metrics, which could be adapted for the metrics we needed. This dApp [**1**] takes input from a user, wraps it into a transaction, and with the help of the SDK it broadcasts it to the network. We updated it to include new GUI buttons and functional actions for retrieving that node's status information. We added buttons tied to actions for testing blacklisting and testing minimum stake. We also needed a way to see the current node's info (metadata, and blacklist status). These would display whether the node is blacklisted, how long the node is blacklisted, reputation, and the node's metadata. Additionally, the tests for blacklisting and minimum stake were implemented in the code of the dApp, tied to those buttons. For both tests, it utilized the blockchain core's messaging system to retrieve a list of all online participants to use for these purposes. Upon clicking the test buttons, the testing would begin and was controlled entirely in the coded structure of the dApp. Meaning, the code would select 25% of the participants from the retrieved list, then send messages as transactions to the core to blacklist or increase their reputation, depending on the test. Then, it would sleep (enough time for the work to be done, with the block added to the chain). After the sleeping pause, it would send out messages for the next 25%, and so on until 100% of participants handled. We also needed to add new programming to the blockchain's engine to handle these new requests, when certain keywords were seen in the transaction note (e.g. blacklist——¡some node's address¿). Further specifics are discussed in the next section.

5.5.2. Testing Process

The goal of our simulations is to show that the modifications which we introduce to the Algorand system do not affect the functionality of the resulting PoRev architecture. In particular, we show that our proposed blockchain system has liveliness and completeness properties. Specifically, blacklisting does not affect liveliness or completeness (that a transaction is guaranteed to be added to the blockchain eventually) when $< 75\%$ of nodes

are blacklisted, and requiring a minimum stake to participate always shows liveliness and completeness regardless of the number of parties not meeting the stake requirement when based upon the rules defined in a previous section.

Our tests were made on a single Dell 8-core system running Ubuntu 18.04 Linux with 16GB of memory. For both tests, our most common system setup started 12 simulated nodes (clients), each in their own console window tab.

Let us now discuss the process we followed for testing. For blacklisting, we want to show that liveliness and completeness are preserved when an increasing number of participants are removed through BL. The test sent three messages (as transactions) in quick succession for each blacklist bracket (0% blacklisted, 25%, 50%, 75%, 100%). When blacklisting was performed (25% and larger brackets), the message contained the note "$blacklist|| < addr >$", which the core interprets as "blacklist this address" and proceeds to call the functionality to do that.

At every bracket, the dApp would submit three more addresses to be blacklisted. When the core added those transactions to the blockchain, it would signal back to the dApp the completion and the dApp would write out the metrics to a file (which node, time submitted to network, time transaction added to a block). This test was run three separate times, and the aggregated outcome is discussed in the next section (see Fig 5.2).

For minimum reputation (stake) testing, the message transactions were done in a similar manner with each bracket (0%, 25%, 50%, 75%, and 100%) representing the percent of nodes with higher reputation than their initial value of one. The message sent as a transaction would be "$minstake|| < addr >$", which the blockchain core would interpret as "increase the reputation of this address by 500" in our tests and then it would run that functionality.

When the core added those transactions to the blockchain, it would signal back to the dApp the completion and the dApp would write out the metrics to a file (which node, time submitted to network, time transaction added to a block). With those metrics and additional data from the node's log (current MinStake, number of eligible nodes to participate), the

results were aggregated and discussed in the next section (see Fig. 5.3).

5.5.3. Metrics and Simulation Results

Our first focus was to study how blacklisting affects the consensus times (the agreement on which a block of transactions are added to the blockchain). We wanted to show that liveliness and completeness in each round is preserved regardless of having nodes blacklisted. Informally, liveliness implies that the consensus round ends in a reasonable amount of time, i.e., the protocol does not get caught waiting indefinitely, and proceeds to the next round. Also informally, completeness implies that a transaction is eventually added to the blockchain, regardless of the number of rounds it takes.

Next, we ran a study to test how a minimum stake ($MinStake$) requirement affects the dynamic reputation and consequently the time it took the system to come to a consensus, while assuring the set of possible participants is never zero. All twelve nodes started with the same reputation value of 10001, and therefore the same stake value calculated from that reputation. Although we assume that initially, the user joins the system with a reputation of 1, for this test, we assumed that we are working with a "bootstrapped" version of the system where the parties already gained a substantial yet equal reputation. This is done in order to save time on running the experiments by ensuring large jumps in reputation values.

In Fig. 5.2, we show that nine messages (as transactions) were submitted for each blacklisting bracket (0% blacklisted, 25%, 50%, 75%, and 100%), tested as a triplet of messages. The first two brackets, 0% and 25%, had similar times of 23-36 seconds. When 50% were blacklisted, the timing doubled to 75-84 seconds. At 75% blacklisted, only the first test run produced completed transactions at 107-113 seconds. With the other tests, it never completed and consequently showed a lack of completeness for transactions when 75% of the nodes in the system are blacklisted. None of the 100% bracket completed, as expected. The system continued adding empty blocks and progressing to the next round. The shape of this plot (and of 5.3), specifically where the message triplets show a pattern of timing decline (e.g. see message 10, 11, 12), is most likely due how time metrics were implemented. It records the time when a message is submitted, and then when it is added to the block.

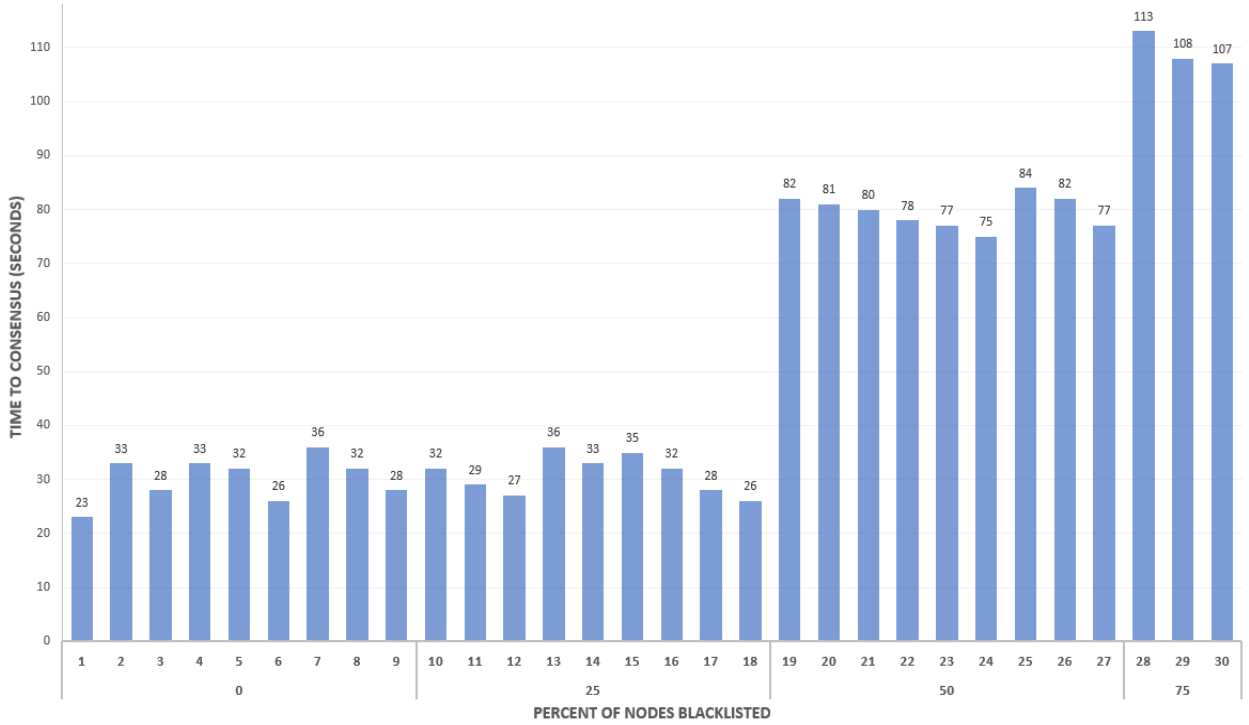

FIGURE 5.2. Blacklisting Nodes vs Time to Consensus.

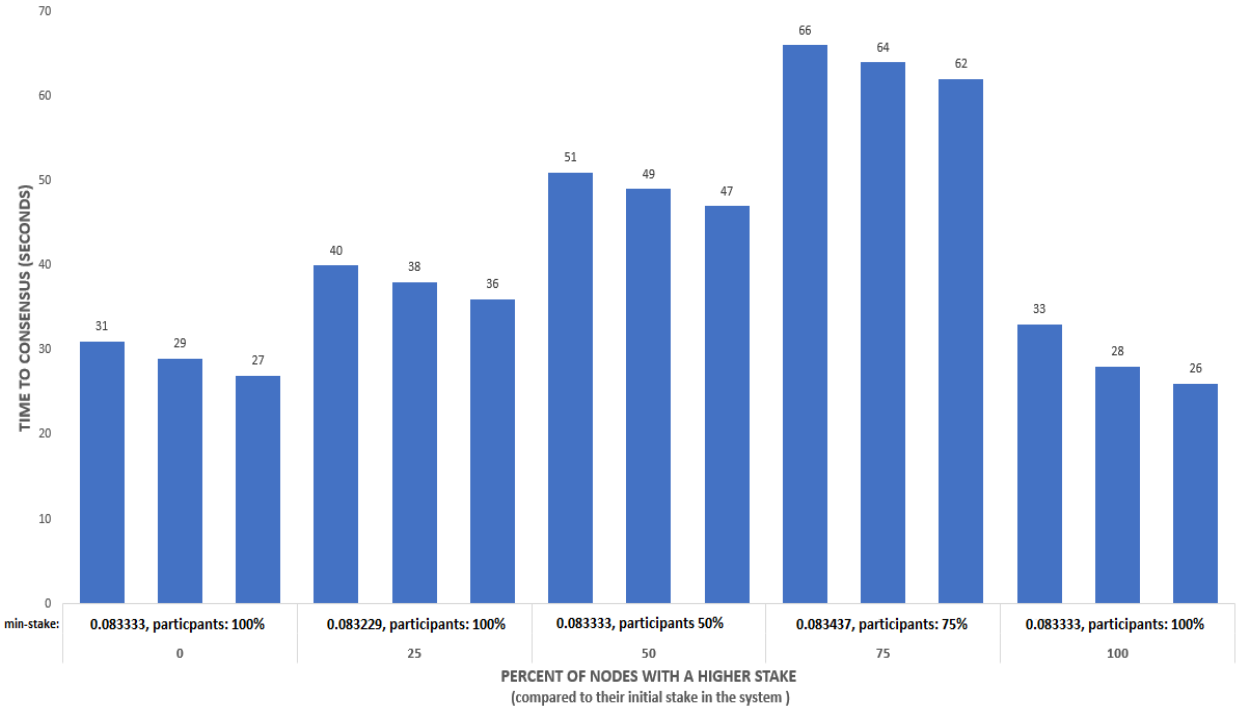

FIGURE 5.3. Minimum Stake's Effect on Consensus Time.
Note: Each bracket is the percent of nodes with a higher reputation than initially.

Since the three messages are added to a block at the same time, but are submitted successively one after the other, it can be deduced that the timing will be different (with the first message trivially longer than the next).

This experiment has shown that blacklisting players for technical or behavioral maliciousness does not affect that round's "honest users agreeing on the same block" or the time interval of when they know of that block (properties from [**16**]). Essentially, blacklisting does not affect liveliness, correctness, or completeness (that a transaction is guaranteed to be added to the blockchain eventually) when $< 75\%$ of nodes are blacklisted.

In Fig. 5.3, we again ran our system with 12 nodes each having equal initial reputation values of 10001. Three messages (as transactions) were submitted to the blockchain system for each reputation bracket (0% with higher reputation, 25%, 50%, 75%, and 100%). At each bracket, three additional nodes had their reputation increased the same amount from their initial reputation value. In the results of the Minimum Stake test, the first two brackets, 0% and 25%, showed similar times for completion, i.e., when a transaction is added to the blockchain. At 50%, the second median of the set of reputation (R) values is greater than the lowest value in R, so 50% of the nodes are excluded from participating in the consensus protocol. In this case, the timing increased to 47-51 seconds. At 75%, the minimum stake has increased and 75% of the nodes meet this stake value at equal or higher levels. The timing jumped again upwards to 62-68 seconds, which could be caused by the amount of time it took to form committees due to there being fewer possible participants available. As expected, when 100% of the nodes have a reputation higher than their initial value, then 100% can participate again (similar to the 0% bracket), since all nodes have the same reputation and stake values again.

5.6. Conclusion

5.6.1. Concluding Remarks

With this new model, Proof of Review (PoRev) we have shown how technology can be used to derive the trustworthiness of both the review and the reviewer participants. It does this through an evaluation of reviews to determine whether it is congruent, trustworthy, and

honest. Additionally, we have shown how our model prevents maliciousness and provides strong security guarantees for the data, participants, and reputation values. We employed blacklisting and a minimum stake requirement. We showed how our Proof of Review model addresses several types of attacks that are common for Proof of Stake and Reputation systems. We implemented the proposed Proof of Review consensus by modifying Algorand's core engine. We also modified an existing dApp to allow us to validate our system as well as to demonstrate the performance of blacklisting and minimum-reputation / minimum-stake components. Since we altered and extended Algorand's engine for our protocol, we examined Algorand's Theorem 1 and associated security properties in respect to our changes. We show in proofs that our modifications effectively continue preserve the security properties.

In testing, our simulation results showed that our proposed blockchain system has liveliness and completeness properties. Specifically, blacklisting does not affect liveliness or completeness (that a transaction is guaranteed to be added to the blockchain eventually) when $< 75\%$ of nodes are blacklisted. We also show that requiring a minimum stake to participate always shows liveliness and completeness regardless of the number of parties not meeting the stake requirement when based upon the rules defined this paper.

5.6.2. Future Work

The following directions are recommended for the future work.

Combining reputation and tokens. This approach will aid in determining a user's weighted influence in a token-based system, with reputation calculated into that weight. This could allow a user with a high reputation and a small amount of tokens to have a greater influence and opportunity to participate in the protocol.

Block re-assembly. During the consensus steps, we may allow the block leader to modify their block by removing review transactions, when the majority of verifiers identify any transactions as "bad". Such transactions can be defined as either technically incorrect or the verifier's evaluation is disparate from the leader's evaluation (including the blacklisting flag for the reveiwer). A one-time cycle is used to give the leader a single opportunity to remove "bad" TXs and reassemble a block of "good" TXs. If the leader fails to remove

them, the verifiers will vote to blacklist the leader; otherwise, the new block is voted to be the round's block.

Application, Micro-accreditation. Exploring ways this model can be leveraged in micro-accreditation (connecting students with employers) to aid in assigning rigor to courses and associated knowledge units. Since computational costs of PoW should be avoided, a PoS system or similar would be beneficial in speeding up transaction times. Proof of Review is an excellent fit, especially when it mitigates some specific pain points in our earlier research. We believe micro-accreditation can be programmed using our new platform.

Application. Other future work includes investigating uses in autonomous systems, like ad-hoc caravanning vehicles and swarm drones. In the context of the discussion on minimum reputation in Section 5.3.5, it is interesting to study the application reputation distributions and the approaches to ensure a reputation-fair lottery.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1. Conclusion

In this work, we investigated solutions for decentralizing trust management processes using blockchain technologies. We have provided some answers to problems with innovative direction and testing. We explored and shown how a blockchain lends itself well to solving these issues in trust by decentralizing the respective architecture. The following is our overall results and conclusions.

6.1.1. Micro-Accreditation Using Blockchain Applications

We first set our focus on accreditation for workforce development, specifically in micro-accreditation. We examined ways to help match employers' knowledge requirements with students' knowledge earned.

The metrics utilized in our analysis of hiring include those reserved for efficiency, scalability, and operational success. The metrics utilized in our analysis of peer-review include gas cost and transaction times, showing the effect of gas price (incentive for the miners to do the work) on the latter. These metrics and their testing methodologies are outlined in Chapter 3.

In order to measure an impact of our proposed solution, we simulated a representative dataset gathered from Amazon's LinkedIn hiring data, which utilize data sets from UC Berkeley, UT Austin, and UNT, as well as rigor rankings from CSRankings.org. In order to gauge efficiency, we measured the amount of gas consumed by each transaction run on the network, taking gas is a measure of the efficiency of a function. To gauge scalability, we measured the amount of gas consumed as the number of entities accessing the network grows—in this case, we considered students. We tracked this using the Ganache emulator and observed how different transactions respond to a high network load. Next, let us discuss the results of our testing in detail.

Simulated and real hiring of Amazon based on data sets.

We showed that the current algorithm is generally capable of distinguishing hiring patterns, and that the new graduates' hiring rates from UT Austin and UC Berkeley are similar to that of the given data set. However, it is observed that the data for UNT in the simulated run is completely zeroed out. The error for this measurement can be attributed to the algorithms extra weight on the rigor of an academic program. Because of this, one can say that the current data set is not capable of properly representing the state of real-world hiring, and because of this, is insufficient to properly deduce the success of the hiring algorithm.

Efficiency of running transactions on the network.

It was observed that the growth of transaction gas cost was generally stable as the number of entities on the network increased. The only transaction whose gas cost grows is the Approve-Credit transaction. This is due to the de-allocation of memory containing the student data structure. Once again, this problem can be fixed through the inclusion of BigChainDB, as most memory-management functions can be handled by BigChainDB. Because of this, we can say that the system is reasonably scalable as well.

Gas cost to deploy School and PeerReview contracts.

Additionally, we found that the gas cost is steep for deploying the School contract for peer-review. The School contract manages all the courses which all have assignments, as well as managing and interacting with the PeerReview contracts when necessary. The School contract has every function that can be transacted on by the user through RPC call from the dApp. There are additional functions (not shown in the graph) used in each library that are only called from another function.

Peer Review transaction times, comparing times when gas price (miner's incentive to do the work) is increased.

Furthermore, we analyzed the transaction times for the peer-review transactions. By default, when you call a transaction, the gas price is 1. The gas price (GP) is the incentive for a miner to work, and the higher the price is, the faster the transaction will happen. Changing the GP from 1 to 50 significantly and positively affected the transaction times. The Gas

Price increase incentivizes the transaction. Manually adjusting the GP using Metamask is laborious, so we changed the transaction call in the code to reflect a specific GP of 50. In the future, testing should be done on the optimal price when needing a more immediate response calling the transaction.

Gas Cost of each function in the Assignment Grading part.

We examined the gas cost for calling the Assignment Grading functions. The 0 (zero) gas costs are Calls, meaning they do not alter any state (i.e., contract variables). The handinAssignment function consumes the most gas and will need to be improved in the future code-changes to significantly lower that cost.

Transaction times for the Assignment Grading.

Finally, we analyzed the transaction times for the Assignment Grading. Similar to the Peer Review times, increasing the GP to 50, significantly lowered the times. In example, at $GP = 1$, the "addAssignment" function times started at about 18 minutes to a much more reasonable 1.5 minutes (approximately). More research is needed to find a good balance for speed and gas price.

From these results presented, we can deduce that the system was:

- Efficient. This is seen in the second metric regarding the transaction gas cost, as all transactions with exception of one transaction, fell beneath a safe threshold. The exception transaction can be fixed in that through implementation of the proposed system, it is possible for one to further optimize the system.
- Scalable. All transactions had constant complexity when varied with network load, except for credit. The scalability of the entire system, with all parts together, suggests possibly needing a different blockchain network requiring the consumption of fewer resources with faster transaction times. Since the cost to incentivize faster transaction times still leaves us with an undesirably slow process, new forms of consensus also may be needed to aid these above goals. Switching to Proof-of-Stake or Proof-of-Review (proposed in Chapter 5 and deploying sharding will be helpful as well.

- Successful (partially). Outside the above scalability issues, success has been demonstrated through our earlier investigation of hiring rates from UT Austin and UC Berkeley, as the relative hiring ratios for the two institutions from the data given was quite close together. However, the algorithm places quite a bit of weight on rigor, which requires further investigation.

Future work includes full implementation of the system by further fine-tuning the algorithm to properly account for rigor scores. This will allow us to further investigate the relationship between rigor of an institution and hiring rates, and further optimize the dApp.

For this research, we initially chose the most popular platform in 2019, Ethereum, with smart contracts. We learned we needed a faster and lower-cost platform to use for the remaining research and decided on proof-of-stake Algorand which had numerous things we needed as its base (including being open-source). We believe micro-accreditation can be programmed using our new PoS-derived platform in the future.

6.1.2. Review Credibility via NLP Analysis on the Blockchain

The next problem we examined is that there is no blockchain mechanism or consensus that exists to ensure that a peer-reviewed rigor score given is honest, unbiased, and trustworthy. We focus our attention on the lack of trust and review credibility in decentralized systems such as online marketplace. We are focused on a particular aspect of trust in reviews and reputation of the parties who provide them: review credibility. Specifically, we are focusing on the following particular scenario, which is the first step towards the above-mentioned mechanism: A party leaves a review (say, on a product or service), which consists of a text and a rating. We used NLP to evaluate the "positivity" of the text, and then compared it to the rating. A trustworthy review is expected to have a good match of the positivity to the rating.

For this problem, our first focus was testing transaction times (or how long it takes for a transaction to be added to a block). In other words, we were investigating whether review transactions are slower than the regular payment transactions. Next, we ran a study on an overhead which the NLP component introduces. These tests are important for showing

that the new review transaction does not affect the timing significantly, and that the system continues to perform well even when review transaction are evaluated by NLP in real time. Next, we tested an accuracy of the NLP evaluation. We expect the NLP technology to perform similarly to what human would deliver.

We compared consensus times for both payment and review transactions. We used a simulated non-live evaluation of reviews for the review transaction. The block consensus times for review transactions range from 40 seconds to 1 minute 52 seconds, with a few outliers above 2 minutes, when using tokens as a stake type in a Proof of Stake system. The timings were similar for both types of transactions. This shows that using different transaction types in a non-live evaluation setting does not affect the overall time for consensus on a block to be added to the ledger.

We then tested block consensus times when using real-time NLP to evaluate reviews using a small dataset adapted from Amazon's data (see Chapter 4). This dataset was handpicked from random product reviews, 2-3 sentences maximum, and placed into a JSON file to be loaded at dApp runtime. Since we are using NLP's resource-heavy sentimental analysis evaluation on three simulated nodes, we limit each block to one review transaction. The block consensus times for real-time NLP range from 49 seconds to 3 minutes and 4 seconds (with one outlier of 4 minutes 26 seconds). This shows that time to consensus can fluctuate when using a real-time evaluation system, with more than 50% of the transactions finishing within the range of cases when NLP is not used (less than 1 minute 52 seconds). This may be caused by either the complexity of the text being evaluated or the resources available during the round. We emphasize that the custom-selected review dataset consisted of comments of about 2-3 sentences, and therefore the real-life timing may differ from the one reported above, depending on a specific dataset.

We note that although the underlying system prefers the quickest times to ensure an accurately credible reputation for each reviewer, the reviewer and potential buyers would not likely care much about the speed that new reviews show up on the front-end.

Finally, we tested the accuracy of the NLP evaluation feature. Our goal is to use

the technology which evaluates reviews in the same way as humans do. This means that if a person perceives a review as mostly positive, then the NLP system should also appraise the same text as mostly positive. The same dataset of product reviews was used as in the previous test. We gauge how positive the review comment is through sentimental analysis, which provides a single number (0-5) evaluation for each sentence, where 0 denotes very negative, and 5 denotes very positive. If this system is provided multiple sentences, it returns an evaluation for each sentence. Our implementation computes an average of all those evaluation scores and then the result is scaled to the interval (0-100). In some cases, the rating matches the evaluation: e.g., see the review text 5, "Trash filled with more trash". In other cases, the rating given by the reviewer does not match the associated text: e.g., see the review text 2, "Best book ever read". In the latter case, the reviewer's rating does not match their comment; however, our system catches this, evaluates the comment accurately, and determines the level of (in)congruence. From this, our system will decide whether the reputation should be increased or decreased.

In conclusion, with this new model we have moved towards showing how technology can be used to evaluate the trustworthiness of both the reviews and the corresponding reviewers. We deploy NLP to determine whether the reviews are congruent and trustworthy.

We note that this information can also be useful at a higher level, such as in dApps for various purposes. For instance, a dApp can rely on this immutable information, using the evaluation scores of the reviews for making decisions. The sole incentive in our model is to increase one's reputation. No financial incentive is currently present, although it may be introduced in the future if potential applications demand it. Our system provides an accurate way to implement an automated analysis of reviews ensuring the trustworthiness of the evaluation.

Our preliminary results show comparable block consensus timings for the cases of using tokens or reputation as a stake in our Proof of Stake component. Additionally, we show that a real-time NLP evaluation may introduce a substantial overhead to about 50% of transactions. This may be caused by either the complexity of the text being evaluated or

the resources available during that round.

More significantly, by using a dataset adapted from Amazon product reviews, it is demonstrated the NLP evaluation component performs similarly to a human evaluation (ground truth). As mentioned in previous sections, our proposal derives the trustworthiness and credibility of a participant via evaluation of their reviews, which is in turn reflected in their reputation. Then, this reputation is used by the consensus algorithm.

6.1.3. Proof-of-Review Consensus

The above research focused on applying this solution to decentralized marketplaces using "off the shelf" components and did not deliver the functionality we would like to have. Our next problem examined was to generalize our research to other problem types to provide the functionality we need. We needed to provide better trustworthiness of both the data stored on a blockchain and the nodes participating in the consensus process with this new functionality. One area we focused on is the length of the effect of a malicious player in the network. Our solution was to implement a new consensus protocol to implement the needed mechanisms to prevent the "bad guys" from participating in the consensus as quickly as possible, when acting both technically and behaviorally malicious. We also investigated ways to better prevent Sybil attacks in a reputation driven system. We implemented our new Proof of Review (PoRev) consensus by heavily modifying the Algorand blockchain engine and system. This allowed us to validate our system as well as to demonstrate the performance of blacklisting and minimum-reputation components.

We needed a way to test our new functionality in an easy and controllable way. We decided on the distributed app (dApp) used from the previous problem, since it had a re-configurable graphical interface and its simplicity to reprogram for testing parts of our protocol. Additionally, this dApp already had a mechanism to output metrics, which could be adapted for the metrics we needed.

The goal of our simulations was to show that the modifications which we introduce to the Algorand system do not affect the functionality of the resulting PoRev architecture. In particular, we show that our proposed blockchain system has liveliness and complete-

ness properties. Specifically, blacklisting does not affect liveliness or completeness (that a transaction is guaranteed to be added to the blockchain eventually) when $< 75\%$ of nodes are blacklisted, and requiring a minimum stake to participate always shows liveliness and completeness regardless of the number of parties not meeting the stake requirement when based upon the rules defined in a previous section.

Our first focus was to study how blacklisting affects the consensus times (the agreement on which a block of transactions are added to the blockchain). We wanted to show that liveliness and completeness in each round is preserved regardless of having nodes blacklisted. Informally, liveliness implies that the consensus round ends in a reasonable amount of time, i.e., the protocol does not get caught waiting indefinitely, and proceeds to the next round. Also informally, completeness implies that a transaction is eventually added to the blockchain, regardless of the number of rounds it takes.

Next, we ran a study to test how a minimum stake requirement affects the dynamic reputation and consequently the time it took the system to come to a consensus, while assuring the set of possible participants is never zero. All twelve nodes started with the same reputation value of 10001, and therefore the same stake value calculated from that reputation. Although we assume that initially, the user joins the system with a reputation of 1, for this test, we assumed that we are working with a "bootstrapped" version of the system where the parties already gained a substantial yet equal reputation. This is done in order to save time on running the experiments by ensuring large jumps in reputation values.

In our results, we show that three messages (as transactions) were submitted for each blacklisting bracket (0% blacklisted, 25%, 50%, 75%, and 100%). The first two brackets, 0% and 25%, had similar times of 23-36 seconds. When 50% were blacklisted, the timing doubled to 75-84 seconds. At 75% blacklisted, only the first test run produced completed transactions at 107-113 seconds. With the other tests, it never completed and consequently showed a lack of completeness for transactions when 75% of the nodes in the system are blacklisted. None of the 100% bracket completed, as expected. The system continued adding empty blocks and progressing to the next round.

This experiment has shown that blacklisting players for technical or behavioral maliciousness does not affect that round's "honest users agreeing on the same block" or the time interval of when they know of that block (properties from [**16**]). Essentially, blacklisting does not affect liveliness, correctness, or completeness (that a transaction is guaranteed to be added to the blockchain eventually) when $< 75\%$ of nodes are blacklisted.

To test minimum stake, we again ran our system with 12 nodes each having equal initial reputation values of 10001. Again, three messages (as transactions) were submitted to the blockchain system for each reputation bracket (0% with higher reputation, 25%, 50%, 75%, and 100%). At each bracket, three additional nodes had their reputation increased the same amount from their initial reputation value. In the results of the Minimum Stake test, the first two brackets, 0% and 25%, showed similar times for completion, i.e., when a transaction is added to the blockchain. At 50%, the second median of the set of reputation (R) values is greater than the lowest value in R, so 50% of the nodes are excluded from participating in the consensus protocol. In this case, the timing increased to 47-51 seconds. At 75%, the minimum stake has increased and 75% of the nodes meet this stake value at equal or higher levels. The timing jumped again upwards to 62-68 seconds, which could be caused by the amount of time it took to form committees due to there being fewer possible participants available. As expected, when 100% of the nodes have a reputation higher than their initial value, then 100% can participate again (similar to the 0% bracket), since all nodes have the same reputation and stake values again.

With this new consensus model, Proof of Review (PoRev), we have shown how this technology can be used to derive the trustworthiness of both the review and the reviewer participants. It does this through an evaluation of reviews to determine whether it is congruent, trustworthy, and honest. Additionally, we have shown how our model prevents maliciousness and provides strong security guarantees for the data, participants, and reputation values. We employed blacklisting and a minimum stake requirement. We showed how our Proof of Review model addresses several types of attacks that are common for Proof of Stake and Reputation systems, as seen in Chapter 5. We implemented the proposed Proof of Review

consensus by modifying Algorand's core engine. We also modified an existing dApp to allow us to validate our system as well as to demonstrate the performance of blacklisting and minimum-reputation / minimum-stake components. Since we altered and extended Algorand's engine for our protocol, we examined Algorand's Theorem 1 and associated security properties in respect to our changes. We show in proofs that our modifications effectively continue to preserve the security properties.

In testing, our simulation results showed that our proposed blockchain system has liveliness and completeness properties. Specifically, blacklisting does not affect liveliness or completeness (that a transaction is guaranteed to be added to the blockchain eventually) when $< 75\%$ of nodes are blacklisted. We also show that requiring a minimum stake to participate always shows liveliness and completeness regardless of the number of parties not meeting the stake requirement when based upon the rules defined this paper.

6.2. Future Work

The following directions are recommended for the future work.

6.2.1. Combining Reputation and Tokens

This approach will aid in determining a user's weighted influence in a token-based system, with reputation calculated into that weight. This could allow a user with a high reputation and a small amount of tokens to have a greater influence and opportunity to participate in the protocol. This could better incentivize participants to "play nice" and do the "right thing", since money is on the line and there would be a small financial reward as is currently in PoS systems.

6.2.2. Block Re-Assembly

During the consensus steps, we may allow the block leader to modify their block by removing review transactions, when the majority of verifiers identify any transactions as "bad". Such transactions can be defined as either technically incorrect or the verifier's evaluation is disparate from the leader's evaluation (including the blacklisting flag for the reveiwer). A one-time cycle is used to give the leader a single opportunity to remove "bad"

transactions and reassemble a block of "good" ones. If the leader fails to remove them, the verifiers will vote to blacklist the leader; otherwise, the new block is voted to be the round's block.

6.2.3. Industrial Applications

Other future work is to explore ways this model can be leveraged in autonomous systems or micro-accreditation (connecting students with employers) to aid in assigning rigor to courses and associated knowledge units. Since computational costs of PoW should be avoided, a PoS system or similar would be beneficial in speeding up transaction times. Proof of Review is an excellent fit. In sensor-array systems, a consensus on a node's assessment being incongruent could indicate a failure or corruption of a sensor. Furthermore, we would also like to explore the application towards autonomous vehicles that caravan together in an ad-hoc manner (consensus on decisions on a route, re-routing if new information, determining speed, assessments to deliver a vehicle safely from point A to point B, etc.). A variation would need to be considered, including a hybrid decentralized-centralized solution. Vehicles would only need a limited blockchain history but still trust in the credibility of others. In the context of the discussion on minimum reputation in Section 5.3.5, it is interesting to study the application reputation distributions and the approaches to ensure a reputation-fair lottery.

REFERENCES

[1] *AlgoChatPoR - Proof of Review dapp*, `github.com/ZeeNexus/algochatPoR`, [Online].

[2] *AlgorandPoR - Proof of Review (core engine modified)*, `github.com/ZeeNexus/algorandPoR`, [Online].

[3] *AlgorandPoR Go SDK - Proof of Review SDK in Go Language*, `github.com/ZeeNexus/go-algorandpor-sdk`, [Online].

[4] *Ganashe - Truffle Suite*, `trufflesuite.com/ganache/`, [Online].

[5] Alchemy, *Ethereum's Ropsten Testnet: A Complete Guide*, `www.alchemy.com/overviews/ropsten-testnet`, [Online].

[6] Ahmed S. Almasoud, Farookh Khadeer Hussain, and Omar K. Hussain, *Smart contracts for blockchain-based reputation systems: A systematic literature review*, Journal of Network and Computer Applications 170 (2020), 102814.

[7] Musab A. Alturki, Jing Chen, Victor Luchangco, Brandon M. Moore, Karl Palmskog, Lucas Peña, and Grigore Rosu, *Towards a verified model of the Algorand consensus protocol in Coq*, Lecture Notes in Computer Science abs/1907.05523 (2020).

[8] Marianne A Azer, Sherif M El-Kassas, Abdel Wahab F Hassan, and Magdy S El-Soudani, *A survey on trust and reputation schemes in ad hoc networks*, Availability, Reliability and Security, 2008. ARES 08. Third International Conference on, IEEE, IEEE, 2008, n/a, p. 881–886.

[9] Shaimaa Bajoudah, Changyu Dong, and Paolo Missier, *Toward a decentralized, trustless marketplace for brokered iot data trading using blockchain*, 2019 IEEE International Conference on Blockchain (Blockchain), 2019, pp. 339–346.

[10] Hanneh Bareham, *Itt tech student loan forgiveness update: What former students need to know*, `www.bankrate.com/loans/student-loans/what-itt-tech-students-need-to-know-about-loan-forgiveness/`, 2023, [Online].

[11] Golam Dastoger Bashar, Joshua Holmes, and Gaby G. Dagher, *Accord: A scalable*

multileader consensus protocol for healthcare blockchain, Trans. Info. For. Sec. 17 (2022), 2990–3005.

[12] Imran Bashir, *Blockchain age protocols*, pp. 331–376, Apress, Berkeley, CA, 2022.

[13] BitBay, *Bitbay decentralized marketplace double deposit escrow*, `bitbay.market/double-deposit-escrow`, 2020, [Online].

[14] Sonja Buchegger, Jochen Mundinger, and Jean-Yves Le Boudec, *Reputation systems for self-organized networks: Lessons learned*, Technology and Society Magazine, IEEE 27 (2008), 41 – 47.

[15] Jing Chen and Silvio Micali, *Algorand*, `https://arxiv.org/abs/1607.01341`, 2017, [Online].

[16] Jing Chen and Silvio Micali, *Algorand: A secure and efficient distributed ledger*, Theor. Comput. Sci. 777 (2019), no. C, 155–183.

[17] D. M. Chiu and Raj Jain, *Analysis of the increase and decrease algorithms for congestion avoidance in computer networks*, Comput. Networks 17 (1989), 1–14.

[18] ChronoBank.io, *Hiring, rebuilt*, `https://chrono.tech/`, 2018, [Online].

[19] Nicholas Diana, Michael Eagle, John Stamper, Shuchi Grover, Marie Bienkowski, and Satabdi Basu, *Data-driven generation of rubric criteria from an educational programming environment*, Proceedings of the 8th International Conference on Learning Analytics and Knowledge (New York, NY, USA), LAK '18, Association for Computing Machinery, 2018, p. 16–20.

[20] Akanksha Dixit, Arjun Singh, Yogachandran Rahulamathavn BSc(Hons), PhD, and Muttukrishnan Rajarajan, *Fast data: A fair, secure and trusted decentralized iiot data marketplace enabled by blockchain*, IEEE Internet of Things Journal PP (2021).

[21] Zulfadzli Drus and Haliyana Khalid, *Sentiment analysis in social media and its application: Systematic literature review*, Procedia Computer Science 161 (2019), 707–714.

[22] Sanjeev Kumar Dwivedi, Mohammad S. Obaidat, Ruhul Amin, and Satyanarayana Vollala, *Decentralized management of online user reviews with immutability using ipfs and*

ethereum blockchain, 2022 International Mobile and Embedded Technology Conference (MECON) (2022), 534–539.

[23] Society for Human Resource Management, `https://www.shrm.org/hr-today/news/hr-news/pages/shrm-benchmarking-report-4,100-average-cost-per-hire.aspx`, 2016, [Online].

[24] Lisa Frye, *The cost of a bad hire can be astronomical*, `https://www.shrm.org/resourcesandtools/hr-topics/employee-relations/pages/cost-of-bad-hires.aspx`, Aug 2019, [Online].

[25] FTC, *Equifax Data Breach*, `www.ftc.gov/enforcement/refunds/equifax-data-breach-settlement`, [Online].

[26] Fangyu Gai, Baosheng Wang, Wenping Deng, and Wei Peng, *Proof of reputation: A reputation-based consensus protocol for peer-to-peer network*, pp. 666–681, 05 2018.

[27] Google, *Google Cloud Natural Language*, `cloud.google.com/natural-language`, [Online].

[28] Pankaj Gupta, Ritu Tiwari, and Nirmal Robert, *Sentiment analysis and text summarization of online reviews: A survey*, 04 2016, pp. 0241–0245.

[29] Thomas Hardjono and Alex Pentland, *Data cooperatives: Towards a foundation for decentralized personal data management*, `https://arxiv.org/abs/1905.08819`, 2019, [Online].

[30] Yao-Chieh Hu, Ting-Ting Lee, Dimitris Chatzopoulos, and Pan Hui, *Hierarchical interactions between ethereum smart contracts across testnets*, Proceedings of the 1st Workshop on Cryptocurrencies and Blockchains for Distributed Systems (New York, NY, USA), CryBlock'18, Association for Computing Machinery, 2018, p. 7–12.

[31] IAD.gov, *CAE-CD knowledge units*, 2018.

[32] Shailak Jani, *An overview of ethereum its comparison with bitcoin*, International Journal of Scientific & Engineering Research Volume 10, Issue 8 (2017).

[33] Yuhao Jiang, Haiguang Wang, and Tianlun Yi, *Evaluation of product reviews based on text sentiment analysis*, 2021 2nd International Conference on Artificial Intelligence and

Information Systems (New York, NY, USA), ICAIIS 2021, Association for Computing Machinery, 2021.

[34] Pankaj Joshi and Anoj Kumar, *A novel framework for decentralized c2c e-commerce using smart contract*, 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2020, pp. 1–5.

[35] Leonard Kleinrock, Rafail Ostrovsky, and Vassilis Zikas, *Proof-of-reputation blockchain with nakamoto fallback*, Progress in Cryptology – INDOCRYPT 2020: 21st International Conference on Cryptology in India, Bangalore, India, December 13–16, 2020, Proceedings (Berlin, Heidelberg), Springer-Verlag, 2020, p. 16–38.

[36] Nicolas Kube, *"daniel drescher": "blockchain basics: a non-technical introduction in 25 steps"*, Financ Mark Portf Manag 32 (2018), 329–331.

[37] Mario Larangeira, *Reputation atnbsp;stake! anbsp;trust layer overnbsp;decentralized ledger fornbsp;multiparty computation andnbsp;reputation-fair lottery*, Information Security and Cryptology – ICISC 2022: 25th International Conference, ICISC 2022, Seoul, South Korea, November 30 – December 2, 2022, Revised Selected Papers (Berlin, Heidelberg), Springer-Verlag, 2023, p. 195–215.

[38] Stefanos Leonardos, Daniël Reijsbergen, and Georgios Piliouras, *Weighted voting on the blockchain: Improving consensus in proof of stake protocols*, 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 2019, pp. 376–384.

[39] ———, *Weighted voting on the blockchain: Improving consensus in proof of stake protocols*, arXiv, 2021.

[40] Lexalytics, *Lexalytics, an NLP feature stack*, `www.lexalytics.com`, [Online].

[41] Min Li, Helen Tang, and Xianbin Wang, *Mitigating routing misbehavior using blockchain-based distributed reputation management system for iot networks*, 2019 IEEE International Conference on Communications Workshops (ICC Workshops), 2019, pp. 1–6.

[42] Xiaoman Li, Qinghua Zhu, Naina Qi, Jinqiu Huang, Yong Yuan, and Fei-Yue Wang,

Blockchain consensus algorithms: A survey, 2021 China Automation Congress (CAC), 2021, pp. 4053–4058.

[43] Katie Lobosco, *Feds pull the plug on itt tech's accrediting agency*, money.cnn.com/2016/09/22/pf/college/itt-accreditor-acics/, 2016, [Online].

[44] Woolf Development Ltd, *Woolf: Building the first blockchain university*, 2018.

[45] CIO Magazine, *The 11 biggest issues it faces today*, 2019.

[46] Finance Magnates, *Buterin: Proof-of-stake ethereum is really no longer so far away*, 2018.

[47] Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky, *The Stanford CoreNLP natural language processing toolkit*, https://stanfordnlp.github.io/CoreNLP, June 2014, [Online], pp. 55–60.

[48] Kashif Mehboob, Junaid Arshad, and Muhammad Khan, *Empirical analysis of transaction malleability within blockchain-based e-voting*, Computers Security 100 (2021), 102081.

[49] metamask, *MetaMask: The Crypto Wallet for Defi*, https://metamask.io/, [Online].

[50] Satoshi Nakamoto, *Bitcoin: A peer-to-peer electronic cash system*, 2008, [Online].

[51] Shmuel Nitzan and Ruth Ben-Yashar, *The optimal decision rule for fixed-size committees in dichotomous choice situations: The general result*, International Economic Review 38 (1997), 175–86.

[52] nltk, *Natural Language Toolkit (NLTK) for python*, nltk.org, [Online].

[53] Patrick Ocheja, Brendan Flanagan, and Hiroaki Ogata, *Connecting decentralized learning records: A blockchain based learning analytics platform*, Proceedings of the 8th International Conference on Learning Analytics and Knowledge (New York, NY, USA), LAK '18, Association for Computing Machinery, 2018, p. 265–269.

[54] Panchalika Pal and Sushmita Ruj, *Blockv: A blockchain enabled peer-peer ride sharing service*, 2019 IEEE International Conference on Blockchain (Blockchain) (2019), 463–468.

[55] Lifang Peng, Zhong Chen, and Qi Li, *Model and method for evaluating creditability*

of c2c electronic trade, Proceedings of the 8th international conference on Electronic commerce: The new e-commerce: innovations for conquering current barriers, obstacles and limitations to conducting successful business on the internet, 2006, pp. 244–249.

[56] Julie Polisena, Martina Andellini, Piergiorgio Salerno, Simone Borsci, Leandro Pecchia, and Ernesto Iadanza, *Case studies on the use of sentiment analysis to assess the effectiveness and safety of health technologies: A scoping review*, IEEE Access PP (2021), 1–1.

[57] Vishnu Prasad Ranganthan, Ram Dantu, Aditya Paul, Paula Mears, and Kirill Morozov, *A decentralized marketplace application on the ethereum blockchain*, 2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC), 2018, pp. 90–97.

[58] Poonam Rani and Rajul Bhambay, *A comparative survey of consensus algorithms based on proof of work*, Emerging Technologies in Data Mining and Information Security (Singapore) (Paramartha Dutta, Satyajit Chakrabarti, Abhishek Bhattacharya, Soumi Dutta, and Vincenzo Piuri, eds.), Springer Nature Singapore, 2023, pp. 261–268.

[59] Paul Resnick and Richard J. Zeckhauser, *Trust among strangers in internet transactions: Empirical analysis of ebay' s reputation system*, The Economics of the Internet and E-commerce, Emerald Group Publishing Limited, vol. 11, 2002, pp. 127–157.

[60] Abiola Salau, Ram Dantu, Kirill Morozov, Kritagya Upadhyay, and Syed Badruddoja, *Multi-tier reputation for data cooperatives*, pp. 253–273, 02 2023.

[61] Abiola Salau, Ram Dantu, and Kritagya Upadhyay, *Data cooperatives for neighborhood watch*, 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 2021, pp. 1–9.

[62] Alexander Schaub, Rémi Bazin, Omar Hasan, and Lionel Brunie, *A trustless privacy-preserving reputation system*, 05 2016, pp. 398–411.

[63] Holberton School, *Holberton school of software engineering*, 2018.

[64] Ayelet Sheffey, *An agency elizabeth warren criticized for exacerbating the student debt crisis and signing off on the 'worst for-profit*

colleges' just lost its federal recognition, www.businessinsider.com/devos-backed-acics-loses-federal-recognition-biden-elizabeth-warren-profit-2022-8, 2022, [Online].

[65] Alchemy Site, *A Complete Guide to Ethereum's Rinkeby Testnet*, www.alchemy.com/overviews/rinkeby-testnet, [Online].

[66] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts, *Recursive deep models for semantic compositionality over a sentiment treebank*, Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (Seattle, Washington, USA), Association for Computational Linguistics, October 2013, pp. 1631–1642.

[67] Hyland Software, *Learning machine*, https://www.learningmachine.com/, 2018, [Online].

[68] Zeenat Tariq, Sayed Shah, and Yugyung Lee, *Speech emotion detection using iot based deep learning for health care*, 2019 IEEE International Conference on Big Data (Big Data), 2019, pp. 4191–4196.

[69] TeachMePlease, *Teachmeplease – educational courses from all over the world*, https://www.teachmeplease.com, 2018, [Online].

[70] Dejan Vujiić, Dijana Jagodić, and Sinia Ranđić, *Blockchain technology, bitcoin, and ethereum: A brief overview*, 2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH) (2018), 1–6.

[71] Shuai Wang, Yong Yuan, Xiao Wang, Juanjuan Li, Rui Qin, and Fei-Yue Wang, *An overview of smart contract: Architecture, applications, and future trends*, 2018 IEEE Intelligent Vehicles Symposium (IV), 2018, pp. 108–113.

[72] Daniel Davis Wood, *Ethereum: A secure decentralised generalised transaction ledger*, https://gavwood.com/paper.pdf, 2014, [Online].

[73] Jie Xu, Cong Wang, and Xiaohua Jia, *A survey of blockchain consensus protocols*, ACM Comput. Surv. (2023), Just Accepted.

[74] Zachary Zaccagni, Ram Dantu, and Kirill Morozov, *Maintaining Review Credibility*

Using NLP, Reputation, and Blockchain, 2022 IEEE 4th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA), 2022, pp. 58–66.

[75] ———, *Proof of Review - Trust Me, It's Been Reviewed*, 2023 5th Blockchain and Internet of Things Conference (BIOTC23), 2023.

[76] Zachary Zaccagni, Aditya Paul, and Ram Dantu, *Micro-Accreditation for Matching Employer E-Hire Needs*, 2019 IEEE International Conference on Blockchain (Blockchain), 2019, pp. 347–352.