Proof of Review: Trust Me, It's Been Reviewed

ZACHARY ZACCAGNI,

Computer Science and Engineering, University of North Texas

ZacharyZaccagni@my.unt.edu, USA

RAM DANTU,

Computer Science and Engineering, University of North Texas

Ram.Dantu@unt.edu, USA

KIRILL MOROZOV,

Computer Science and Engineering, University of North Texas

Kirill.Morozov@unt.edu, USA

We present a novel consensus model called Proof of Review (PoRev) which uses reputation, evaluation of reviews/assessments, blacklisting, and minimum reputation stake to come to an agreement on a block of transactions, while securing the data on the blockchain. PoRev is a tool that can be used to aid applications by ensuring honest and unbiased reviews and assessments. The motivation to develop this is for applications in decentralized marketplaces, micro-accreditation, community-based blockchain applications like a neighborhood watch, and possibly for caravanning autonomous vehicles. Our proposed architecture is based on Algorand, where we heavily modified the engine to implement our consensus. The reviews and evaluations are stored on the blockchain, driving the reputation model, where reputation is used to regulate a user's participation in the system—similarly to Proof of Reputation systems. A player may be blacklisted and prevented from participating in the protocol, if agreed on 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. Minimum stake is employed and is chosen in a way to prevent malicious new users access to participate, 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 (therefore stake) to participate beyond submitting reviews.

Our simulation results showed that the proposed blockchain system has liveliness and completeness. Specifically, blacklisting and the minimum-reputation requirement (when properly tuned) do not affect liveliness or completeness (that a transaction is guaranteed to be added to the blockchain eventually).

CCS Concepts: • **Computing methodologies → Distributed computing methodologies**; • **Computer systems organization → Distributed architectures**; • **Security and privacy → Distributed systems security**.

Additional Key Words and Phrases: blockchain, consensus protocol, proof of review, review credibility, reputation

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. 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 (consensus routes, re-routing, speed, safety actions, etc.). 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 [17], 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. 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 [4], 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 [14]. *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 [18, 20]. 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 [17] and similar ones such as data cooperatives—see the references in [17]). 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. [9], 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).

1.1 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. [2] 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. [11] 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. [16] 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. [12] 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 [13] 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. [18] use reputation as a stake in the context of data cooperatives [10]. The idea of using reputation to incentivize the parties' honest behavior—which originates from earlier works such as [9]—is employed in our construction as well. However, we apply it in a different context of evaluating review credibility. Also, the work [18] uses a different blockchain system, the Snow White protocol. Zaccagni et al. [20] introduce a model to prevent bad reviews through combining off-the-shelf elements of reputation, evaluation, and the Algorand blockchain systems. The authors show that using NLP with a blockchain can be used to ensure trust in the reviews in a decentralized marketplace. We also will be using a reputation and evaluation system, but we are modifying the engine to implement our Proof of Review consensus, by generalizing the evaluation system and adding additional functionality like blacklisting and minimum stake.

1.2 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 [20]. 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.

[1]We note that the minimum stake handling in Algorand does not quite satisfy our goals. See Section 3.5 for discussion.

Instead of using Algorand in the off-the-shelf model—as it is done in [20]—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 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.

The rest of this paper is organized as follows. Section II briefly describes the components of the system, including the blockchain engine. Section III introduces our proposed architecture and its implementation. Section IV presents the security analysis. Section V discusses our simulation results, and concluding remarks are made in Section VI.

2 BACKGROUND

In this section, we briefly discuss the PoS-based Algorand blockchain system and its mechanics, and we refer the interested reader to [4] 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 [4]. 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 [4] 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 [3] (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 [1]. 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 [4] 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.

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.

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 like working with reputation (instead of tokens) and a evaluation mechanism (we chose to use an NLP system like [20] 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. 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. 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.

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 [18, 20]. 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

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

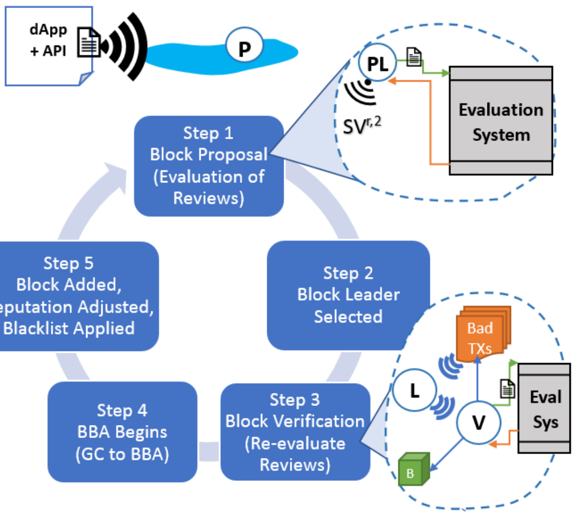

Fig. 1. Steps of Proof of Review.

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 in calculating weighted votes during consensus as

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

is the weighted proportional stake, 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.

The Public Ledger is extended. 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. 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.

A Review Transaction (*ReviewTX*) that does not require payment is introduced and is derived from PaymentTX defined in [3]. This is similar to the review transaction in [20], 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 [20] 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 $SV^{r,s} \neq \emptyset$.

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 [20] 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.

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. 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. Methods and communication to and from Step 4 Verifiers remain the same as Algorand.

Step 3: 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.

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 acting 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 acting 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.

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. [12] to resolve the issue.

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 [4].

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 [4]. 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 [4].

Theorem 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 [4].

At this point, we can borrow the Completeness Lemma (Lemma 5.3 of [4]), which is stated as follows.

Lemma 1 (Completeness Lemma). Assume Properties 1–3 of Theorem 1 hold for rounds $0, \ldots, r-1$. When the leader ℓ^r is honest, Properties 1 and 2 of Theorem 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 III-E and Figure 1), 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 [4] will still hold taking into account those changes.

Lemma 2 (Soundness Lemma). Assume that Properties 1–3 hold for rounds $0, \ldots, r-1$. 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 [4], 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 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][4] 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 CM19, 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 5.2 from [4].

Lemma 3 ([4]). *Assuming that Properties 1-3 of Theorem 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 3, Theorem 1 follows by induction (similarly to [4]).

4.1 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 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.

5 SIMULATION RESULTS

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 3, and the resulting is made available in [6]. A new metadata struct was associated with each node (see section 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 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 [20], 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 [8] 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.

5.2 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 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. 3).

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

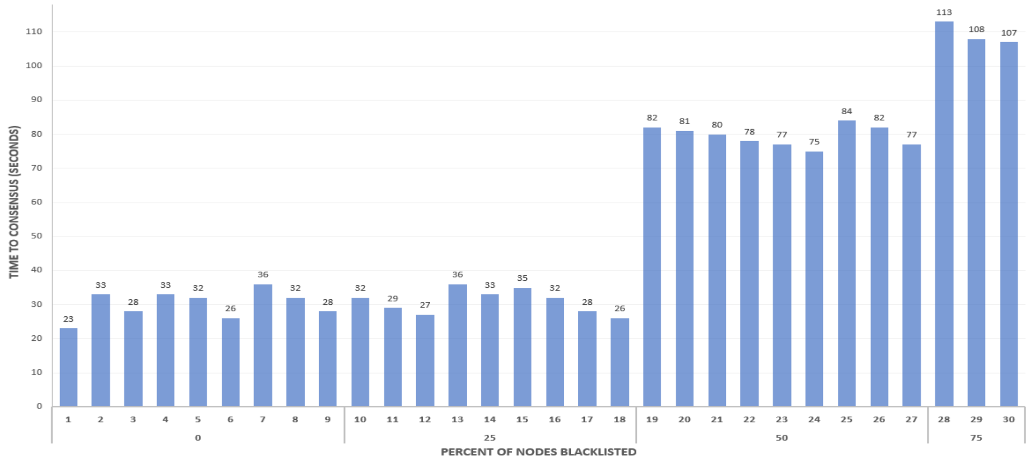

Fig. 2. Blacklisting Nodes vs Time to Consensus.

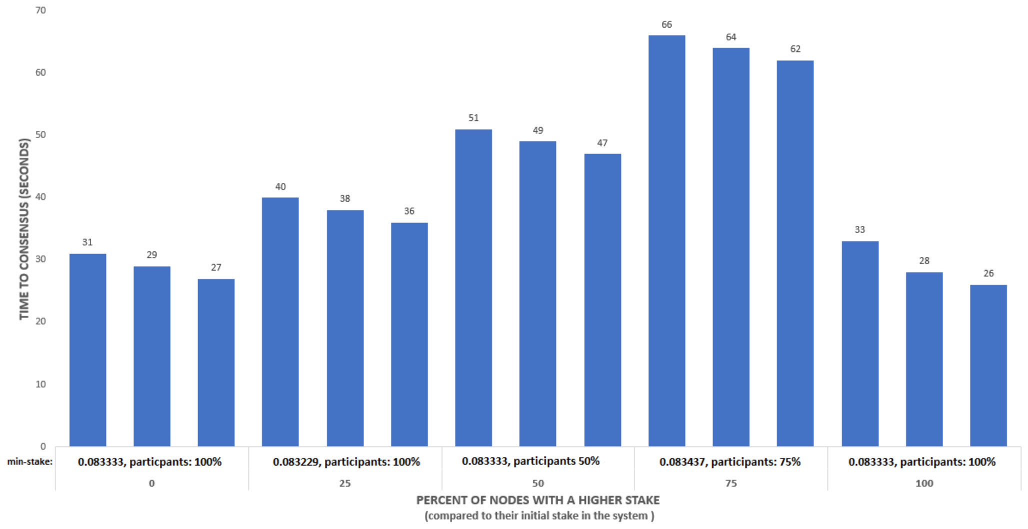

Fig. 3. Minimum Stake's Effect on Consensus Time. Each Bracket is the Percent of Nodes with a Higher Reputation than Initially

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. 2, 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 [4]). 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. 3, 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.

6 CONCLUSION

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.

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. 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. In the context of the discussion on minimum reputation in Section 3.5, it is interesting to study the application reputation distributions and the approaches to ensure a reputation-fair lottery.

REFERENCES

[1] M. Alturki, J. Chen, V. Luchangco, B. Moore, K. Palmskog, L. Pena, G. Rosu, "Towards a verified model of the algorand consensus protocol in coq", Lecture Notes in Computer Science, Volume 12232 LNCS, 2020, Pages 362-367

[2] G. D. Bashar, J. Holmes and G. G. Dagher, "ACCORD: A Scalable Multileader Consensus Protocol for Healthcare Blockchain", IEEE Transactions on Information Forensics and Security, Volume 17, 2022, Pages 2990-3005

[3] J. Chen, S. Micali, "Algorand" ArXiv abs/ abs/1607.01341 (2017).

[4] J. Chen, S. Micali, "Algorand: A secure and efficient distributed ledger", Theoretical Computer Science, Volume 777, 19 July 2019, Pages 155-183, https://doi.org/10.1016/j.tcs.2019.02.001.

[5] L. Cheng, J. Liu, C. Su, K. Liang, G. Xu, W. Wang, "Polynomial-based modifiable blockchain structure for removing fraud transactions", Future Generation Computer Systems, Volume 99, 2019, Pages 154-163

[6] Github AlgorandPoR github.com/ZeeNexus/algorandPoR

[7] Github Go AlgorandPoR SDK github.com/ZeeNexus/go-algorandpor-sdk

[8] AlgoChatPoR github.com/ZeeNexus/algochatPoR

[9] F. Gai, B. Wang, W. Deng, and W. Peng, "Proof of Reputation: A Reputation-Based Consensus Protocol for Peer-to-Peer Network", Database Systems for Advanced Applications, Springer International Publishing, 2018, pp. 666–681

[10] Hardjono, T., and Pentland, A. (2019). Data cooperatives: Towards a foundation for decentralized personal data management. arXiv preprint arXiv:1905.08819.

[11] K. Khan, J. Arshad, M. Khan, "Empirical analysis of transaction malleability within blockchain-based e-Voting", Computers & Security, Volume 100, 2021

[12] L. Kleinrock, R. Ostrovsky, V. Zikas, "Proof-of-Reputation Blockchain with Nakamoto Fallback", Progress in Cryptology – INDOCRYPT 2020, Lecture Notes in Computer Science, vol 12578, 2020

[13] M. Larangeira, "Reputation at Stake! A Trust Layer over Decentralized Ledger for Multiparty Computation and Reputation-Fair Lottery", Proc. of 25th International Conference on Information Security and Cryptology (ICISC 2022), pp. 195–215, Springer, 2023.

[14] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system", bitcoin.org/bitcoin.pdf

[15] QuantumMechanic, "Proof of stake instead of proof of work", Bitcoin Talk site, bitcointalk.org/index.php?topic=27787.0

[16] A. Schaub, R. Bazin, O. Hasan, L. Brunie, "A Trustless Privacy-Preserving Reputation System" 2016, 398-411 10.1007/978-3-319-33630-5_27

[17] Salau, A., Dantu, R., & Upadhyay, K. (2021). Data Cooperatives for Neighborhood Watch. 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 1-9.

[18] A. Salau, R. Dantu, K. Morozov, K. Upadhyay, S. Badruddoja, "Multi-Tier Reputation for Data Cooperatives", 2022, MARBLE

[19] C. Smith, Other Ethereum Authors, "Proof-of-stake (PoS)", 2022, ethereum.org/en/developers/docs/consensus-mechanisms/pos/

[20] Z. Zaccagni, D. Dantu, K. Morozov, "Maintaining Review Credibility Using NLP, Reputation and Blockchain", IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS), 2022

[21] Q. Zhuang, Y. Liu, L. Chen, and Z. Ai, "Proof of Reputation: A Reputation-based Consensus Protocol for Blockchain Based Systems", 1st International Electronics Communication Conference (IECC '19), ACM, 2019