A PoR/PoS-Hybrid Blockchain: Proof of Reputation with Nakamoto Fallback

Leonard Kleinrock
UCLA

Rafail Ostrovsky
UCLA

Vassilis Zikas
University of Edinburgh

Abstract

Reputation is a major component of trustworthy systems. In this work, we describe how to leverage reputation to establish a highly scalable and efficient blockchain. In order to avoid potential safety concerns stemming from the subjective and volatile nature of reputation, we propose a proof-of-reputation/proof-of-stake-hybrid (in short, PoR/PoS-hybrid) blockchain design. Although proof-of-stake and proof-of-reputation have been separately studied, to our knowledge, our proposal is the first cryptographically secure design of proof-of-reputation-based (in short PoR-based) blockchains; and it is the first blockchain that fortifies its PoR-based security by optimized Nakamoto-style consensus. This results in a ledger protocol which is provably secure if the reputation system is accurate, and preserves its basic safety properties even if it is not, as long as the majority of the stake in the system remains in honest hands. Our results put emphasis on reputation fairness as a key feature of any reputation-based lottery. We devise a definition of reputation fairness that ensures fair participation while giving chances to newly joining parties to participate and potentially build reputation. We also describe a concrete lottery in the random oracle model which achieves this definition of fairness. Our treatment of reputation-fairness can be of independent interest.

Keywords: Blockchain, cryptocurrencies, proof of reputation, proof of stake

1 Introduction

Many of the decisions taken in modern society are based on centralized publicly known reputation systems. For example, fans are likely to follow suggestions from their idols, social network followers are likely to adopt suggestions of the friends and groups they follow. Most importantly, people use publicly available ranking systems such as E-Bay, Yelp, AirBnB, Amazon, etc to take decisions regarding online-shopping, choice of vacation, accommodation, eating out, insurances, investment, medical, etc.

In this work we leverage the power of reputation systems to establish a reliable, permissionless blockchain. Our design assumes that certain parties in the system have a public "reputation" similar to E-bay or Yelp, which is recorded on the blockchain itself. The more "stars" a party has, i.e., the higher its ranking, the better is this party's reputation. And because reputation can be destroyed by misbehaving, the higher the reputation the higher are the chances that the parties will behave honestly in a decentralized consensus protocol execution.

In a nutshell, our design goals are two-fold: (1) to rely on the reputation system to build a simple, highly scalable decentralized ledger with optimized finality, communication, and computation, and with a formal proof that relates the quality of our system to the quality of the reputation system; (2) to back our ledger's safety with a fall-back mechanism that ensures that even if the reputation estimate is flawed, the ledger protocol does not create indefinite forks.

1.1 Our Results

At the core of most blockchain-based decentralized consensus protocols is an implicit lottery that designates, for each block, one or more parties with the right to propose the next block in the blockchain, along with a mechanism to endorse this proposal so that consensus is achieved across the blockchain networks. We propose a provably secure Proof-of-Reputation/Proof-of-Work-hybrid (in short, PoR/PoS-hybrid) blockchain.

Our blockchain protocol proceeds in synchronous rounds and relies on a reputation system to select a set of *leaders* (parties that will propose the contents of the next block) and *endorsers* (parties that vote on the next block) that advance the blockchain. In particular, reputation is used to instantiate the above lottery mechanism and achieve decentralized consensus with optimal finality, a concept which is hereafter referred to as *proof of reputation*, in short *PoR*. In the following we sketch the basic design principles for our PoR blockchain.

We assume that we have a (dynamically updatable) set $\mathcal{P}$ of parties where a subset $\hat{\mathcal{P}} \subseteq \mathcal{P}$ of them are special parties called *reputation parties*. The parties wish to leverage the reputation of the reputation parties to maintain a ledger containing a sequence of blocks, each block containing a collection of messages that can represent arbitrary data (throughout this paper, we refer to this data as *transactions*). We do not specify here how this data is efficiently encoded into a block, e.g., so that they can be updated and addressed in an efficient manner; however, one can use the standard Merkle-tree approach used in many common blockchains, e.g., Bitcoin [Nak08, BMTZ17], Ethereum [But13], Cardano/Ouroboros [KRDO17, BGK$^+$18], Algorand [GHM$^+$17], etc.

Informally, a reputation system for a party set $\mathcal{P}$ is similar to a characterization of a probabilistic adversary structure [GIOZ17]: It assigns to each subset $\mathcal{P}$ a probability that this subset is corrupted by the adversary—we only consider active, aka Byzantine, corruption. In its simplest form, which we refer to as *static correlation-free*, a reputation system can be described by a vector of $|\mathcal{P}|$ independent boolean random variables. The probability that the ith variable is 1 is the probability that the i-th party in $\mathcal{P}$—in some canonical ordering, e.g., using the party's IDs—is honest. We note in passing that this corresponds to the notion of independent reputation systems investigated in [ALZ13]. We extend the above notion by allowing the reputation to evolve as rounds advance, yielding a dynamic reputation system. The update happens in an epoch-based manner, where an epoch consists of a fixed number of rounds.

To capture feasibility of PoR-based consensus, we introduce the notion of a *feasible* reputation system (cf. Definition 6) which for static reputation systems requires that there is a party sampling algorithm, such that the probability that a super-majority of the selected parties is honest is overwhelming. (This coincides with the feasibility condition from [ALZ13].)

We use the reputation system in the following manner (we restrict the discussion here to static correlation-free reputation): The contents of the genesis block—which is assumed to be available to any party joining the protocol and includes the (initial) reputation system along with a random nonce—are hashed to generate randomness for the first epoch's lottery. Since every party is assumed access to the genesis block, every party can locally run a lottery which is biased by the parties' reputation using these coins to choose a committee for each slot. The *slot committee* $\mathcal{C}^i_{\text{BA}}$ of the i-th slot[1] is responsible for gathering all known transactions at that slot that have not yet been inserted in the blockchain, and proposing the block corresponding to this slot along with evidence

[1] As discussed below, our description assumes in each slot blackbox access to (a functionality for) Byzantine Fault Tolerant (BFT) consensus; hence when this is instantiated by a BFT consensus protocol, each slot will span over multiple rounds, depending on the underlying BFT protocol. We remark that although we present our protocol assuming access to a deterministic termination canonical functionality [CCGZ16, CCGZ17], we can directly use the compiler from [CCGZ17] to replace this with the constant-round randomized BA protocol (cf. Appendix D.4).

to allow all parties to agree on this block (see details below).

Concretely, in every slot, every party in a random subset $\mathcal{C}^i_{\mathrm{BC}}$ of $\mathcal{C}^i_{\mathrm{BA}}$ of slot leaders pools all new and valid—where validation is done by a predicate which is a parameter of the ledger—transactions received by the blockchain users by that slot into a block, and communicates this to $\mathcal{C}^i_{\mathrm{BA}}$, who run an honest majority BFT consensus protocol to agree on the next block. The agreed upon block is then signed by all members of $\mathcal{C}^i_{\mathrm{BA}}$ and circulated to the party set $\mathcal{P}$ who accept it if and only if it has at least $|\mathcal{C}^i_{\mathrm{BA}}|/2$ signatures from parties in $\mathcal{C}^i_{\mathrm{BA}}$. Clearly, as long as for each of the slot committees the majority of its members is honest (which will be induced by an accurate and feasible reputation system), the above BFT consensus protocol will achieve consensus on the current block (and only this agreed upon block might be accepted by the parties). Once this happens the next slot starts.

In order to allow for adversaries that have partial adaptivity (cf. Appendix 6), we use ideas from the adaptive and/or mobile adversaries literature and introduce a stricter feasibility condition which require a fraction of 3/4 honest parties in each committee. In particular, first we embed in the protocol a lottery re-randomization mechanism: In each round, parties include in their proposed transactions a random nonce which is also incorporated to the contents of the block; in each subsequent epoch, the above process is invoked to select round committees with the difference that the randomness for the lottery is extracted by hashing the contents from the previous epoch. Second, the parties refresh their key in each epoch; this can be done interactively, by posting a new key on the blockchain in every epoch and erasing the previous key, or non-interactively by using a forward secure signature scheme [BM99, IR01]. For simplicity, throughout this paper we focus in our analysis on static reputations. The adaptation to dynamic, i.e., epoch-resettable, adversary is in Appendix D.2.

Of prime importance in our construction, and a novelty of our PoR-lottery, is a mechanism which ensures an appropriate version of egalitarianism. In particular, as proved in [ALZ13], for any feasible correlation-free reputation system, the following sampler, denoted as $\mathtt{A}_{\max}$, outputs such a committee: Order the parties according to their reputation and choose a sufficient number (cf. [ALZ13]) of reputation parties with the highest reputations.

The above suggest that for the type of systems we are discussing here, i.e., correlation-free reputations, the above simple $\mathtt{A}_{\max}$ algorithm is optimal (up to negligible quantities) for minimizing the risk of dishonest majority. Nonetheless, in practice $\mathtt{A}_{\max}$ suffers from several deficiencies: First, if some malicious party establishes (by playing initially honestly) a high reputation, then this party will be included in almost every committee. Second, the approach lacks a natural *fairness* quality which would give parties voting power according to their reputation (even to parties with low reputation). Such a fairness property is instrumental for sustainable decentralization, as it will incentivize parties to join and/or increase their reputation, and, in parallel, it will allow parties that are not in the top reputation ranks to participate in the process, thereby also improving their own reputation.

We propose an appropriate notion of reputation fairness, termed *PoR-fairness*, which addresses the above concerns, along with a delicate lottery mechanism that achieves it. The idea behind PoR-fairness is that every reputation party should get a chance to (be part of the committee that) provides a recommendation for the inclusion of the next block to be added on the blockchain. The challenge is that we want to achieve this while ensuring that, the higher the reputation, the better the relative chances are of a party to be included in the committee. This will incentivize parties to try to increase their reputation (or preserve their current reputation), and allow us to choose many highly reputable parties thereby eliminating the chances of a majority of corrupt reputation parties.

Intuitively, a reputation-based committee selection protocol is *(approximately) fair*, if, (1) on

average, the representation of parties on the selected committee becomes higher, the less likely those parties are to be corrupt (according to their reputation); (2) this representation increases as the ratio of parties with higher over parties with lower reputation increases; and (3) the probabilities of parties included in the lottery increase proportionally to their reputation. Realizing such a reputation-fair lottery turns out to be a challenging goal and a core technical contribution of our work.

Last but not least, our blockchain construction addresses the subjective nature of reputation systems—e.g., inaccuracies due to the fact that the reputation estimator might be flawed and/or dishonest parties might behave honestly until they build sufficient reputation to attack. In particular, we build into the protocol a fallback mechanism based on a Nakamoto-based blockchain. (For the sake of this submission we use a proof-of-stake Nakamoto-style blockchain, e.g., [KRDO17], but our fallback uses the Nakamoto blockchain in a blackbox manner and can therefore be instantiate using any blockchain that realizes a Bitcoin-style transaction ledger [BMTZ17]).This fallback mechanism fortifies the basic security properties of the blockchain under an independent backup assumption, e.g., majority of honest stake. Note that the fallback blockchain is not used for communicating messages, but only digests (hashes) of blocks from the main, reputation chain. Furthermore, it does not need to run in a synchronized manner with the main chain. This allows a very light-weight use of the fallback chain, which as it is a black-box use, can even be outsourced to an existing Nakamoto-style chain. We refer to this type of blockchain as a *PoR/PoS-Hybrid Blockchain*. We note that such a hybrid approach takes advantage of the benefits we can harvest from proof-of-reputation discussed in the following section, without (ab)using massive external resources as is the case with proof-of-work based solutions, e.g., the Bitcoin. Furthermore, by using a blockchain secure in the dynamic availability setting of [BGK$^+$18], our fallback also addresses possible issues with availability of parties in the PoR-blockchain.

1.2 Properties of our construction

The proposed reputation-based blockchain combines and improves several properties of existing constructions:

1. *High Throughput/Transaction-Liveness:* Existing solutions adopt the proposal of at most one effective slot-leader in each slot [KRDO17, DGKR18, BGK$^+$18, GHM$^+$17]. Instead, in our solution a small committee $\mathcal{C}_{\text{BC}}$ of proposers is chosen in each slot, and we allow all of them to include the transactions they see in the next block by communicating them to the larger random committee of slot block-endorsers $\mathcal{C}_{\text{BA}}$. This means that transactions do not need to propagate through the entire network in order to be included, and it suffices that they reach at least one party in the next round's $\mathcal{C}_{\text{BC}}$ committee. This also means that the propagation of transactions is expected to put less load on the network, as once a transaction settles on a block it is removed from the local transaction pool of any party that sees this block and is no longer forwarded by that party. Finally, although the set of parties in $\mathcal{C}_{\text{BC}}$ is small (of size $L = O(1)$, e.g., $L = 5$) for efficiency purposes, still with probability at least $1/2^L$ at least one of the parties in this set will be honest—the reason is that we are choosing L parties out of $\mathcal{C}_{\text{BA}}$ which we know has honest majority—and therefore all transactions this party knows will be included. In contrast, existing solutions both in the proof-of-work and the proof-of-stake realm, e.g., Bitcoin, Ethereum, Cardano, Algorand, etc, adopt the view of a single slot leader, hence transaction liveness is slowed down each time this single slot leader is corrupted.

2. *Efficiency and Scalability:* A major advantage of reputation-system-based constructions is that it limits the adaptivity of the adversary and incentivizes honest (looking) behavior, i.e.,

one does not need to take costly measures to avoid worst case adaptive corruption. Indeed, the chances that a party gets corrupted are defined by the reputation-system. This allows us to avoid the need for heavy adaptive security tools, e.g., compute and communicate proofs of verifiable random functions and, instead, rely on the random oracle model to extract the randomness for the next slot's committee (see also Section D.3 for details on this randomness extraction mechanism.) Nonetheless, our blockchain addresses concerns about adaptivity in corruptions, through its fallback mechanism, which can be adaptively secure.

In addition to the above efficiency benefits, our protocol allows reputation parties to have public IDs, e.g., public IP address, and therefore in each round only the members of the consensus committee $\mathcal{C}_{\mathrm{BA}}$ send or receive messages needed for consensus, and only members of the (even smaller) broadcast committee $\mathcal{C}_{\mathrm{BC}}$ multicast to the whole node set, and only one message each. This allows us to avoid overloading the network with gossiping any communicated message. In addition to reducing the amount of flooding/gossiping, which is expected to have a practical impact on the scalability of the protocol, our protocol yields also an asymptotic improvement on the message complexity, as communication occurs only among the (sublinear) size slot committees and between these committees and the player set. Thus even ignoring the overhead of gossiping, the overall message complexity per slot is $\omega(n^2)$ (in fact, $O(n \log^{1+\epsilon} n)$ for committees of size $\log^{1+\epsilon} n$) as opposed to the $O(n^2)$ complexity of standard flooding/gossiping solutions.

3. *Finality:* Since the parties decide on the next block by means of a traditional consensus protocol, this decision is instant. This is in contrast to standard Nakamoto-style blockchains [Nak08, But13, KRDO17] which achieve eventual consistency, aka the common prefix property [GKL15, PSs17].

The cost for some of the above properties is that, as in Algorand, the reputation parties need to be online when they are chosen. However, this is another place where the reputation system is instrumental. In particular as discussed in the following section, we can fortify our protocol with a volunteering mechanism where parties are rewarded (by increasing their reputation) for volunteering and participating.[2] Parties who wish to preserve and improve their reputation need to volunteer and be online when they do. This mechanism can be even more effective when reputation is linked to an actual ranking systems, such as Yelp or Ebay, where parties also have external incentive structures to preserve and improve their reputation. Thus, the reputation system can also serve as a replacement or reinforcement of standard coin-based reward/punishment mechanisms.

1.3 Related Literature

The recent explosion of blockchain and decentralised ledgers solutions makes it infeasible to review all the related literature here. In the following, we discuss works which are most relevant to our results and treatment. Wherever appropriate, additional protocols relevant to our result are discussed in the following sections.

Reputation systems were previously considered in the context of multi-party computation by Asharov et al. [ALZ13]. Their results establish necessary and sufficient conditions on the reputation distribution of a static reputation system for the existence of fair MPC—in particular, for the existence of an algorithm for selecting a committee where the majority of the participants is honest.

To our knowledge, ours is the first work which puts forth a rigorous specification, protocol, model, and treatment of reputation-system-based blockchains. Attempts to combine consensus

[2] We refer to Section D.5 for a longer discussion of the above mechanisms.

with reputation were previously made in the context of blockchains and cryptocurrencies. Many of these works lack a complete protocol specification, security model and proofs, and often even a credible security argument [GWDP18], and/or rely on complicated reputation mechanisms and exogenous limitations on the adversary's corruption power [Cho07]. Alternative approaches, use the proof-of-work (bitcoin) paradigm to assign reputation, by considering miners more reputable if they allocate, over a long period, more hashing-power to their protocol [YKDE19].

Notably, [BFK17] proposed a reputation-module which can build a scalable blockchain on top of a BFT-style consensus protocol, e.g., PBFT or Honey Badger [MXC+16]. The idea is that this reputation module can be used by the parties to select smaller next round committees; however, in addition to lacking a security proof, the entire module needs to operate over a broadcast channel created by the original BFT consensus protocol, as it uses a global view of the computation to accurately readjust the reputations. Hence, its security relies on the security of the underlying consensus protocol, even if reputation is accurate. Furthermore, their protocol cannot tolerate epoch-resettable adversaries.

Instead our PoR-blockchain construction is secure under the assumptions of accuracy of the reputation system, irrespective of the properties of the fallback blockchain. We note in passing that [BFK17] also proposed an alternative notion of reputation-fairness. Using our terminology, their definition renders a reputation-based lottery more fair the closer its outcome is to a uniform sample. We believe that this simplistic definition is not the right notion of reputation-fair lotteries, as it is unclear why low distance from uniform is a desirable property. (Why should it be considered fair that a large set of parties with low reputation has better relative representation in the output than a small set with higher reputation?) Our fairness definition addresses this concern, at a very low overhead.

We remark that none of the above works addresses the subjective nature of the reputation systems, i.e., if the reputation system is inaccurately estimated, their security fails. This is in contrast to our fallback guarantee which allows us to preserve basic safety (unforkability) properties which are essential in blockchains and cryptocurrencies.

The idea of hybrid constructions which use an alternative consensus mechanism as a fallback was also previously used in Thunderella [PS18] and Meshcash [BHMN17]. Their protocols rely on smart refinements of the proof-of-work and/or proof-of-space-time paradigms, and uses novel tricks to accelerate the blockchain and improve scalability and finality when a higher amount of the underlying resource is in honest hands while ensuring safety even under weaker guarantees. Finally, Afgjort [MMNT19] devises a finality layer module on top of a proof-of-stake blockchain. Their construction achieves instant finality under the combination of the assumptions underlying the PoS-blockchain—typically, honest majority of stake—and the assumption supporting the security of the finality layer. In contrast, our PoR/PoS-hybrid blockchain is secure as long as the reputation-system is accurate irrespective of the security of the underlying PoS-blockchain.

1.4 Organization of the Remainder of the Paper

In Section 2 we establish basic terminology and notation. In Section 3 we lay out the basic parameters of the model. Section 4 provides the basic definition of a reputation-fair lottery and construction assuming a feasible correlation-free reputation system. Finally Section 5 describes a pure PoR-based blockchain for static reputation systems and Section 6 describes how to add a PoS backup leading to our final solution. Due to space limitations some detailed proofs have been moved to the appendix, and are referred to as necessary. The Appendix also includes a discussion about extensions and open research directions.

2 Preliminaries

We use the standard definition of negligible and overwhelming: A function $\mu : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for any polynomial $p(k)$: $\mu(k) = O(1/p(k))$; We say that a function $f : \mathbb{N} \to [0, 1]$ is *overwhelming* if $f(k) = 1 - \mu(k)$ for some negligible function μ. Many of our statements and definitions assume an (often implicit) security parameter k. For two strings $s_1, s_2 \in \{0,1\}^*$ we denote by $s_1||s_2$ the concatenation of s_1 and s_2. For some $n \in \mathbb{N}$ we will denote by $[n]$ the set $[n] = \{1, \ldots, n\}$. For a string $s \in \{0,1\}^k$ and for some $D \leq k$ we will say that $T(s) \geq D$ if s has at least D leading zeros, i.e., s is of the form $s = 0^D||s'$ for some $s' \in \{0,1\}^{k-D}$.

3 The Model

Our blockchain-ledger protocol is a decentralized protocol among a set of nodes $\mathcal{P} = \{P_1, \ldots, P_n\}$, also referred to as *parties*. Each of these parties corresponds to what is typically referred to as a *full node* in the blockchain literature.[3] That is, each node stores a complete and validated view of the blockchain and participates in maintaining it whenever the node is online. We note that the set $\mathcal{P}$ is not static, i.e., new nodes might be added or deleted to the set, and nodes might go temporarily offline, under the restrictions of *semi-dynamic availability* discussed below. A subset of the nodes—which might also be dynamically updated during the protocol's lifetime—have a distinguished role in the protocol, and are in charge of proposing the blocks to be added in the blockchain by a mechanism based on their public reputation (see Section 3.1 for our formulation of reputation). In the remainder of this section we outline the basic network, cryptographic, and setup assumptions used in our construction.

We assume a standard cryptographic adversary who gets to *actively* corrupt parties—i.e., takes full control over them—and uses them to attack the decentralized protocol. The adversary is assumed to be able to perform any efficient computation—i.e., can compute any probabilistic-polynomial time (PPT) algorithm.

Synchrony. Similarly to [BMTZ17, BGK+18] the parties are assumed access to a synchronized clock that allows them to keep track a global current round, so that no (honest) party proceeds to round $\rho + 1$ before all parties have had a chance to complete round ρ. (We discuss how this assumption can be relaxed using techniques from the literature in Appendix D.4.) As a worst case assumption, the adversary is assumed to be *rushing*, i.e., in every round, the adversary first receives all messages that are sent by honest parties, and then decides which messages the corrupted parties should send (if any).

As is the case in standard PoS-based blockchains [BGK+18, GHM+17], synchrony will allow us to have a well structured blockchain timeline which is divided into *slots*, such that in each slot, blocks tagged with that round are added to the blockchain. Furthermore, we will divide time (hence also the blockchain) into epochs of a predefined length, i.e., each epoch consisting of a predefined number of rounds.

The Network. Parties have two means of communication. The first network is a diffusion (multicast) network build by means of a standard gossiping protocol over an incomplete but connected communication graph of unicast channels, whereby a party stores messages, also called *transactions*, it has seen for a fixed time interval and forwards any newly seen transaction to

[3] Here we do not distinguish between full and light nodes but they can easily be added to the system analogous to standard blockchains.

its neighbors. This is analogous to all well studied and widely deployed blockchains such as Bitcoin [Nak08, BMTZ17], Ethereum [But13], Cardano/Ouroboros [KRDO17, BGK+18], Algorand [GHM+17], Thunderella [PS18], etc. For sake of clarity, we will abstract this communication by means of a zero-delay *multi-cast* primitive (cf [BMTZ17]): When an honest party multicasts a message, this message is delivered to all (honest) parties in the network by the beginning of the following round.[4] However, using [CCGZ17] our protocol can be compiled to be secure in the more relaxed setting where the delivery of messages can be adversarially delayed by a bounded amount (see discussions in Appendix D.4.)

The second communication medium is for cases where the identity (e.g., IP) of the receiver is known. This is a direct channel to the receiver (e.g., established over TCP/IP). These will be channels to the *reputation-parties* which have a special role in the protocol—these parties will be part of a reputation system—and might have publicly announced IP addresses which can be used by anyone for directly communicating with these parties in a faster manner than via the vanilla diffusion network. We stress that existence of such links is not necessary for our security analysis—if they are there, they can be used otherwise, communication via the diffusion network is sufficient for security. However, if they exist and are used—as due to the nature of reputation systems, such communication links will exist in reality—then they will considerably reduce the traffic flowing through the network yielding a more scalable/efficient solution, as discussed in Section 1.2.

3.1 Reputation Systems

A *reputation system* $\mathtt{Rep}$ for $m = O(k)$ *reputation parties* from a set $\hat{\mathcal{P}} = \{\hat{P}_1, \ldots, \hat{P}_m\}$ is a family of probability distributions (parameterized by m) over binary *reputation vectors* of length m, i.e., vectors of the type $(h_1, \ldots, h_m) \in \{0,1\}^m$.[5] Each h_i is an indicator bit which takes the value 1 if $\hat{P}_i$ is honest and 0, otherwise. For example, $\Pr[(h_1, \ldots, h_m) = (0, \ldots, 0, 1)] = 0.6$ means that with probability 0.6 every reputation party except $\hat{P}_m$ is dishonest. In this work we consider two types of reputation systems:

- A *static* reputation system is a probability distribution as discussed above. This is similar to the reputation system considered in [ALZ13].
- A *dynamic* reputation system instead is a sequence (ensemble) $\mathtt{Rep} = \{\mathtt{Rep}_\rho\}_{\rho \in \mathbb{N}}$ of distributions, where each $\mathtt{Rep}_\rho$ is a static reputation system for a set $\hat{\mathcal{P}}_\rho$ of $m_\rho \in \mathbb{N}$ reputation parties. The reason for considering such dynamic reputation systems is that in a dynamic blockchain setting, reputation parties might get added or removed, and the reputation of existing parties might change depending on their behavior in the system and/or other exogenous factors—see Appendix D.5 for a discussion. Hence the interpretation of $\mathtt{Rep}_\rho$ is that it corresponds to the snapshot of the reputation system in round ρ. A typical scenario, is a reputation system updated at the beginning of each epoch—i.e., for all rounds ρ and ρ' that belong to the same epoch, $\mathtt{Rep}_\rho = \mathtt{Rep}_{\rho'}$—by means of public information recorded on the blockchain during the previous epoch.

Correlation-free reputation systems. In our description, for simplicity, we focus on the setting where each h_i corresponds to the output of an *independent* indicator random variable H_i, i.e.,

[4] Observe that the adversary might send a message to a subset of parties, but if any honest party is instructed by the protocol to forward it, then the message will be delivered (to all other honest parties) once this forwarding occurs.

[5] For notational simplicity, we often refer to $\mathtt{Rep}$ as a probability distribution rather than an ensemble, i.e., we omit the explicit reference to the parameter m.

whether or not a reputation party $\hat{P}_i$ behaves honestly does not depend on what other reputation parties do. In this case, a static reputation system can be described by a vector of m numbers between 0 and 1, i.e., $\mathtt{Rep} = (\mathtt{R}_1, \ldots, \mathtt{R}_m) \in [0,1]^m$, where the interpretation of $\mathtt{Rep}$ is that with probability equal to $\mathtt{R}_i$ the party $\hat{P}_i$ will play *honestly* (i.e., $\Pr[H_i = 1] = \mathtt{R}_i$).[6] We then say that $\mathtt{R}_i$ *is the reputation of party* $\hat{P}_i$. We refer to such a reputation system as a *correlation-free* reputation system. In the following we provide more details of the corruption capabilities that a correlation-free reputation system (static or dynamic) gives to the adversary. Correlated reputation systems (with limited correlations) and corresponding stronger adversaries are discussed in Appendix D.2.

Corruption of Reputation Parties The adversary's corruption capabilities are specified by the reputation system. We consider three types of *reputation-bounded* adversaries:

- A *fully static* reputation-bounded adversary for reputation system $\mathtt{Rep}$, also referred to as a $\mathtt{Rep}$-*adversary*, corrupts the set of parties at the beginning of the protocol and sticks to this choice. This is the weakest form of corruption considered here. Given a reputation system $\mathtt{Rep}$ for m reputation parties, corruption with a static adversary occurs as follows: A vector $(h_1, \ldots, h_m) \in \{0,1\}^m$ is sampled according to the distribution defined in $\mathtt{Rep}$, and each $h_i = 0$ the reputation party $\hat{P}_i \in \hat{\mathcal{P}}$ is corrupted by the adversary.

- Fully static reputation-bounded adversaries are unsuitable for reputation systems which might be dynamically updated as the underlying system evolves—e.g., new reputation parties are added, and the reputation of existing parties is updated as a result of their (mis)behavior. An *epoch-resettable* reputation-bounded adversary instead, corrupts a completely new set of parties at the beginning of each epoch, according to the reputation system at the beginning of the epoch, and is a more suitable (and stronger) adversary for dynamic reputation systems. The process is similar to the above, with the difference that a new set of corrupted parties is sampled at the beginning of each round ρ for which $\mathtt{Rep}_\rho \neq \mathtt{Rep}_{\rho-1}$.

- The above epoch-resettable adversary is suitable for correlation-free reputation systems. However, this corruption mode might be considered overly restrictive as it does not allow the adversary to "lock onto" corrupted parties across epoch boundaries—i.e., keep a corrupted party corrupted when the epoch switches. In Appendix D.2 we discuss a refinement of this type of adversaries to account for such more general corruption-patterns. This refinement implicitly captures several adversarially-induced dependencies in the distribution of corrupted parties, and, therefore, also captures limited correlations or inaccuracies in the parties' reputation which are still tolerable. Although in this work, security of our PoR-blockchain is proved for correlation-free systems, in Appendix D.2 we discuss how this treatment can be extended using the above refinements to reputation systems with limited correlation. We stress however that, as we prove, even when our reputation system is inaccurate (e.g., there are stronger correlations than allowed in our treatment) the Nakamoto-blockchain (e.g., PoS-blockchain) backup will ensure a sufficiently strong security fallback property.

3.2 Semi-Dynamic Availability

In modern blockchain protocols, parties are allowed to come and go at will. In [BGK$^+$18] the concept of dynamic availability was introduced to capture arbitrary, even adversarial, participation patterns. Instead, in [GHM$^+$17], these patterns were implicitly reduced to assume the necessary

[6] Adaptive correlation-free reputation systems are described, analogously, as an ensemble of static reputation systems.

level participation. Although the advantage of full dynamic availability is evident—i.e., one does not need additional assumptions about which parties are around—this generality does not come without a cost: Protocols with dynamic availability [BGK+18] do not achieve instant finality—i.e., a new block needs to be confirmed by sufficiently many follow-up blocks, which might even arrive at different times, in order to be considered stable. Our design aims to achieve instant finality, i.e., once an honest party adopts a new block, every honest party will adopt it in the same round assuming full synchrony, or by a delay which is bounded by the network's delay if not. This is analogous with the guarantee achieved by Algorand and other iterated Byzantine fault tolerant constructions.

Although, for clarity, our security proof assumes such an assumption on availability guarantees—i.e., that parties are there when needed—we can use reputation to ensure that a rational party which wants to preserve or increase its reputation will participate when necessary. More precisely, we can take the following approach. (Recall that the protocol execution is divided into epochs.) In each epoch, the reputation parties are allowed to declare their availability for up to `E` epochs in the future (where `E` is a protocol parameter). Note that the quality of the blockchain heavily depends of the availability of reputation parties. Therefore, we use the following mechanism to incentivize such availability:

1. The reputation of a party increases with the number of epochs, for which it declares itself available. If this number is less than a set threshold, then the reputation decreases. This will incentivize the reputation parties to declare dense availability.[7]

2. The reputation of a party decreases if the party has declared itself available and is selected to participate but it does not.

Additional reputation-based rewards and penalties are discussed in Section D.4.

3.3 Cryptographic Assumptions and Setups

We conclude this section by providing the setup assumptions used in the specification and proof of our blockchain. As discussed below, some of these assumptions are introduced to simplify the exposition and can be removed using standard techniques from the blockchain literature.

– Random Oracle. This is the standard assumption that a hash function behaves as a random function.

– Randomness Beacon (Source of Clean Randomness). We will assume that the parties have access to a randomness beacon that produces fresh, unpredictable randomness. At the beginning of each round ρ, the beacon generates a fresh independent k-bit random string R_ρ; at any round ρ, parties can query the beacon with any past-round index $\rho' \leq \rho$ and they receive $R_{\rho'}$ as a response. A query for a future round is always answered with $\perp$. We note that such a beacon is not necessary in the simple case of a static or epoch-resettable adversary, where we can use the random oracle to generate randomness for the whole protocol, but it becomes relevant for the more advanced adversary able to tune his corruptions and/or for correlated reputation systems. Therefore we already include it in the original description.

  We note further that, even for the above stronger adversarial assumptions, this beacon assumption is only for simplifying the description. In Section D.3 we discuss how to instantiate a sufficiently strong version of the beacon using a simple and very efficient construction; in particular, we show how the parties can choose the randomness of each epoch by evaluating a

[7] Observe that here we implicitly assume that parties wish to increase their reputation. Exact rewarding mechanisms that realize this incentives structure are discussed in Section D.

hash function (i.e., random oracle) in a hash-chain manner, i.e., on input the randomness of the previous epoch—or the genesis block if this is the first epoch—and (a hash) of the contents of the blocks of the previous epoch. We note that this allows the adversary to "grind" the actual randomness [KRDO17, GHM+17]—i.e., tune the randomness by sampling a different input polynomially many times. However, as each epoch will also include honest inputs—transactions, blocks, and nonces—the input will have sufficient entropy to ensure unpredictability of the next epoch's randomness.

– Genesis Block. We assume that parties are able to obtain the genesis block which includes sufficient randomness. In practice, this randomness can be extracted from a combination of unpredictable yet publicly verifiable events, e.g., recent news articles and contents, or from other existing blockchains.

– Complexity Assumptions. In terms of cryptographic assumptions we will assume existentially unforgeable digital signatures [GMR88] along with pseudorandom function. Efficient and even post-quantum variants of these primitives are known to exist under standard complexity assumptions, namely, existence of one-way functions or hardness of lattice problems.

4 Reputation-based Lotteries

In the heart of our construction is a lottery that chooses a (sublinear) set of parties according to their reputation. To demonstrate the idea, let's first consider two extreme scenarios:

- Scenario 1: No reputation party $\hat{P}_i$ has $\mathtt{R}_i > 0.5$.
- Scenario 2: All reputation parties $\hat{P}_i$ have (independent) reputation $\mathtt{R}_i > 0.50000001$.

In Scenario 1, one can prove that users cannot use the recommendation of the reputation parties to agree on a sequence of transactions. Roughly, the reason is that with good probability, the majority of the reputation parties might be dishonest and try to split the network of users, so that they accept conflicting transaction sequences. Thus in this scenario, consensus needs to be reached by leveraging some other mechanism. We outline such a mechanism based on proof of stake in Section 6.

In Scenario 2, on the other hand, the situation is different. Here, by choosing a small random committee of, say 40, reputation parties, we can guarantee (except with negligible probability)[8]that the majority of those parties will be honest (recall that we assume independent reputations), and we can therefore employ a fast Byzantine Fault-tolerant Consensus (BFT) protocol to achieve agreement on the sequence of transactions.

Definition 1. *For a reputation system* Rep *for parties from a reputation set* $\hat{\mathcal{P}}$*, a (possibly probabilistic) algorithm* A *for sampling a subset of parties from* $\hat{\mathcal{P}}$*, and an* Rep*-adversary* $\mathcal{A}$*, we say that* Rep *is* $(\epsilon, \mathtt{A})$-feasible for $\mathcal{A}$ *if, with overwhelming probability,*[9] A *outputs a set of parties such that at most a* $1/2 - \epsilon$ *fraction of these parties is corrupted by* $\mathcal{A}$*.*

Note that in the above definition, the corrupted parties are chosen according to Rep from the entire reputation-party set $\hat{\mathcal{P}}$, and independently of the coins of A. (Indeed, otherwise it would be trivial to corrupt a majority.)

[8] All our security statements here involve a negligible probability of error. For brevity we at times omit this from the statement.

[9] The probability is taken over the coins associated with the distribution of the reputation system, and the coins of $\mathcal{A}$ and A.

Definition 2. *We say that a reputation system is ϵ-feasible for* Rep*-adversary $\mathcal{A}$, if there exists a probabilistic polynomial-time (PPT) sampling algorithm* A *such that* Rep *is $(\epsilon, \mathtt{A})$-feasible for $\mathcal{A}$.*

It is easy to verify that to maximize, in the above Scenarios, the (expected) number of honest parties in the committee it suffices to always choose the parties with the highest reputation. In fact, as show in [ALZ13] this can be generalized to arbitrary correlation-free reputation systems. In particular, [ALZ13] proved that for any ϵ-feasible reputation system Rep (i.e., for any Rep-adversary $\mathcal{A}$) the algorithm which orders that parties according to their reputation chooses sufficiently many (see. [ALZ13]) parties with the highest reputation induces a set which has honest majority. We denote this algorithm by $\mathtt{A}_{\max}$.

Lemma 1 ([ALZ13])**.** *A correlation-free reputation system is ϵ-feasible for a* Rep*-adversary $\mathcal{A}$ if and only if it is $(\epsilon, \mathtt{A}_{max})$-feasible for $\mathcal{A}$.*

We note that as discussed in the introduction, although yielding a maximally safe lottery, $\mathtt{A}_{\max}$ has issues with fairness which renders it suboptimal for use in a blockchain ledger protocol. In the following we introduce an appropriate notion of reputation-fairness for lotteries and a mechanism for achieving it.

4.1 PoR-Fairness

Analogous to standard ranking/reputation systems, we divide reputation in a small number of tiers. This is analogous to the ranking system used by classical reputation/recommendation systems, e.g., in Yelp, a party might have reputation represented by a number of stars from 0 to 5, along with their midpoints, i.e., 0.5, 1.5, 2.5, etc—we refer to such a reputation system as a *eleven-tier reputation-system*. As a warm up we first informally demonstrate fairness and describe a reputation-fair lottery for the simple case of a two-tier reputation system. In the following paragraph we provide our formal definition of reputation-fairness along with a sampler for a constant number of reputation tiers.

4.1.1 A Simple PoR-Fair Lottery for a 2-Tier Reputation System

For the purpose of this demonstration let us assume that all reputations can be divided in two tiers corresponding to reputation at least 0.75000001 and reputation at least 0.50000001 (but lower than 0.75000001). This will allow us to focus on fairness as all candidates for the committee-lottery have reputation greater than 0.50000001 which, as discussed above, will ensure (with overwhelming probability) honest majority within the committee. The more general case of more tiers and tiers with lower probabilities is discussed in the following paragraph.

For the purpose of this example, we denote the set of reputation parties with reputation between 0.50000001 and (less than) 0.75000001 by $\hat{\mathcal{P}}_2$ and the set of parties with reputation at least 0.75000001 by $\hat{\mathcal{P}}_1$. Assume that $|\hat{\mathcal{P}}_1| = \alpha$ and $|\hat{\mathcal{P}}_2| = \beta$. We want to sample a small number y of parties in total, where, of course, $y < |\hat{\mathcal{P}}| = |\hat{\mathcal{P}}_1| + |\hat{\mathcal{P}}_2|$. Per our specification of the tiers, the parties in $\hat{\mathcal{P}}_2$ are about twice more likely to be corrupted as parties in $\hat{\mathcal{P}}_1$. Hence, if the sets $\hat{\mathcal{P}}_1$ and $\hat{\mathcal{P}}_2$ were of similar size, the fairness requirement would imply a goal of selecting (on average) about twice as many parties from $\hat{\mathcal{P}}_2$ than the number of parties selected from $\hat{\mathcal{P}}_1$. However, when $|\hat{\mathcal{P}}_1| = \alpha_1$ is much larger than $|\hat{\mathcal{P}}_2|$, we want to start shifting the selection towards $\hat{\mathcal{P}}_1$, as we do not want to penalize large sets of high-reputation parties—high reputation is a desideratum. This leads to the following informal goal for the lottery (the formal definition is discussed in the next paragraph).

Goal (informal): We want to randomly select x_1 parties from $\hat{\mathcal{P}}_1$ and x_2 parties from $\hat{\mathcal{P}}_2$ so that:

1. $x_1 + x_2 = y$, and
2. $x_1 = \max\{2, \frac{\alpha_1}{\alpha_2}\}x_2$.
3. For each $i \in \{1, 2\}$: No party in $\hat{\mathcal{P}}_i$ has significantly lower probability of getting picked than other parties in $\hat{\mathcal{P}}_i$, but parties in $\hat{\mathcal{P}}_1$ are twice as likely to be selected as parties in $\hat{\mathcal{P}}_2$.

Here is how to create a lottery which achieves the above goal. To avoid having to deal with rounding issues, we choose our target size y to be such that the values $\frac{2\beta-\alpha}{\alpha+\beta}\frac{y}{1+\frac{2\beta-\alpha}{\alpha+\beta}}\alpha = \frac{5\beta-\alpha}{3\beta}\alpha y$ and $\frac{y}{1+\frac{2\beta-\alpha}{\alpha+\beta}}\beta = \frac{\alpha+\beta}{3}y$ are integral values. The lottery proceeds as follows:

1. First, we choose $\max\left\{0, \frac{2\beta-\alpha}{\alpha+\beta}\right\}\frac{y}{1+\max\left\{0, \frac{2\beta-\alpha}{\alpha+\beta}\right\}}$ parties randomly from $\hat{\mathcal{P}}_2$;
2. Subsequently, we choose $\frac{y}{1+\max\left\{0, \frac{2\beta-\alpha}{\alpha+\beta}\right\}}$ parties randomly from the union $\hat{\mathcal{P}}_1 \cup \hat{\mathcal{P}}_2$. If a party from $\hat{\mathcal{P}}_2$ is chosen twice, i.e., in both of the above steps, then choose another party randomly from the parties in $\hat{\mathcal{P}}_2$ that have not yet been chosen in any of the steps.
3. Take all the parties that were chosen in the above steps to form the committee.

We next show that the outcome of the above lottery satisfies, on average, the goals set for it. (**Notation:** For a random variable X, we denote by $\mathbb{E}(X)$ the expected value of X). The non discrimination property follows from the fact that the parties are chosen randomly from their respective sets. As a warm up, the following theorem formalizes and proves the fact that, "on average", our above lottery satisfies the first two properties discussed in the above goal. The formal and more general version of the result, with all three properties, is then proved in the following paragraph for arbitrary (feasible) reputation systems:

Theorem 1 (Reputation-Fair Lottery for two tiers)**.** *In the above lottery, for $i \in \{1, 2\}$: let X_i be the random variable (r.v.) corresponding to the set of parties in the final committee that come from set $\hat{\mathcal{P}}_i$ (i.e., $|X_i|$ is the r.v. corresponding to x_i.) Let also ℓ_I (resp. ℓ_{II}) denote the number of parties that are added on the committee in the first step (resp. in the last two steps). Then the following statements hold:*

1. $\ell_I + \ell_{II} = y$
2. $\mathbb{E}(|X_1|) = \max\{2, \frac{\alpha}{\beta}\}\,\mathbb{E}(|X_2|)$

The proof of the theorem can be found in Appendix B.1.

4.1.2 Extension to Multiple Reputation Tiers

We next discuss how to formalize the above informal fairness requirement, and extend the definition and lottery mechanism to more than two reputation tiers and to allow reputations less than 1/2. Note that although, in principle, a reputation less than 1/2 indicates that the respective reputation party is more likely to be adversarial than not, it will be essential for the dynamic reputation version of the protocol to select (even a small number) of such parties, to give newly joining reputation parties—which, being new, might have lower reputation—a chance to build reputation by honestly playing the protocol. Along the way we also describe the formal specification of a sampler for achieving our notion of fairness.

To this direction we will partition the reputations in $m = O(1)$, i.e., constantly many, tiers as follows: For a given small $\delta > 0$, Tier j includes parties with reputation between $\frac{j-1}{m} + \delta$ and $\frac{j}{m} + \delta$. Note that we explicitly ignore parties with reputation 0; those parties are not considered as belonging to the reputation set. Thus, a mechanism for removing a reputation party is to set its reputation to 0.

We refer to the above partitioning of the reputations as an *m-tier partition.* The idea of the fairness goal for the lottery then is to give to each party belonging to Tier j, chances of being chosen proportional to $c^{(m-j)}$ for an appropriate class of constant c's, which are parameters to the the sampling algorithm, but also balanced by the tier-set size ratio, similarly to the two-tier case. Concretely, the formal fairness goal is as follows:

Definition 3. *Let $\hat{\mathcal{P}}_1, \ldots, \hat{\mathcal{P}}_m$ be a partition of the reputation-party set $\hat{\mathcal{P}}$ into m subsets and let L be a lottery which selects x_i parties from each $\hat{\mathcal{P}}_i$. For some $c \geq 1$, we say that L is c-*reputation-fair*, or simply, c-*fair *if it satisfies the following properties:*

1. *(c-Representation Fairness): For $j = 1, \ldots, m$, let $c_j = \max\{c, c \cdot \frac{|\hat{\mathcal{P}}_j|}{|\hat{\mathcal{P}}_{j+1}|}\}$. Then L is c-*fair *if for each $j \in \{0, \ldots, m-1\}$ and for every constant $\epsilon \in (0, c)$:*

$$\Pr\left[(c_j - \epsilon)\, x_{j+1} \leq x_j \leq (c_j + \epsilon)\, x_{j+1}\right] \geq 1 - \mu(k),$$

   for some negligible function μ.

2. *(c-Selection Fairness): For any $p_j \in \cup_{i=1}^{m} \hat{\mathcal{P}}_i$, let $\textsc{Member}_i$ denote the indicator (binary) random variable which is 1 if p_j is selected by the lottery and 0 otherwise. The L is c-selection-fair if for any $i \in \{1, \ldots, m-1\}$, for any pair $(p_{i_1}, p_{i_2}) \in \hat{\mathcal{P}}_i \times \hat{\mathcal{P}}_{i+1}$, and any constant $c' < c$:*

$$\frac{\Pr[\textsc{Member}_{i_1} = 1]}{\Pr[\textsc{Member}_{i_2} = 1]} \geq c' - \mu(k)$$

   for some negligible function μ.

3. *(Non-Discrimination): Let $\textsc{Member}_i$ defined as above. The L is non-discriminatory if for any p_{i_1}, p_{i_2} in the same $\hat{\mathcal{P}}_i$:*

$$\textsc{Member}_{i_1} \approx \textsc{Member}_{i_2},$$

   where $\approx$ in the above equation means that the random variables are computationally indistinguishable.

At a high level the lottery for the m-tier case is similar in spirit to the two-tier case: First we sample a number of α_1 parties from the highest reputation set, then we sample α_{12} parties from the union of second-highest and the highest, then we sample α_{123} parties from the union of the three highest reputation tiers, and so on. The values $\alpha_1, \alpha_{12}, \alpha_{123}$, etc. are carefully chosen so that the above fairness goal is reached whenever there are sufficiently many parties in the different tiers. We next discuss the detailed mechanism and prove its security properties.

We start by describing two standard methods for sampling a size t subset of a party set $\mathcal{P}$—where each party $P \in \mathcal{P}$ is associated with a unique identifier *pid*[10]—which will both be utilized in our fair sampling algorithm. Intuitively, the first sampler samples the set with replacement and the second without. The first method, denoted by Rand, takes as input/parameters the set $\mathcal{P}$, the size of the target set t—where naturally $t < |\mathcal{P}|$—and a nonce r. It also makes use of a hash function

[10]In our blockchain construction, *pid* will the P's public key.

h which we will assume behaves as a random oracle.[11] In order to sample the set, for each party with ID *pid*, the sampler evaluates the random oracle on input (pid, r) and if the output has more than $\log t$ tailing 0's the party is added to the output set. By a simple Chernoff bound, the size of the output set $\bar{\mathcal{P}}$ will be concentrated around t. For completeness we include a description of the sampler along with its achieved properties in Appendix A

In addition to the above sampler which samples each party independently, we will also make use of the straighforward sampler, denoted by $\mathsf{RandSet}$ (see Appendix A), that samples a random subset of t parties from $\mathcal{P}$ (corresponding to sampling without replacement): Order the parties according to the output of h on input (pid, r) and select the ones with the highest value (where the output h is taken as the standard binary representation of integers). It follows directly from the fact that h behaves as a random oracle—and, therefore, assigns to each $P_i \in \mathcal{P}$ a random number from $\{0, \ldots, 2^k - 1\}$—that the above algorithm uniformly samples a set $\bar{\mathcal{P}} \subset \mathcal{P}$ of size t out of all the possible size-t subsets of $\mathcal{P}$.

Theorem 2 (Reputation-Fair Lottery for $m = O(1)$-tiers)**.** *In the above lottery* $\mathsf{L}(\hat{\mathcal{P}}, \mathtt{Rep}, (c_1, \ldots, c_m), \delta, \epsilon, r)$*, let* $\epsilon, \delta > 0$ *be strictly positive constants, and for each* $i \in \{1, \ldots, m = O(1)\}$*, let* X_i *be the random variable (r.v.) corresponding to the set of parties in the final committee that come from set* $\hat{\mathcal{P}}_i$ *(i.e.,* $|X_i|$ *is the r.v. corresponding to* x_i*); and let* $c_i = \max\{c, c\frac{|\hat{\mathcal{P}}_i|}{|\hat{\mathcal{P}}_{i+1}|}\}$ *where* $c = O(1)$ *such that for some constant* $\xi \in (0,1) : \frac{1}{c^{m-1}} \leq \frac{m-2}{2m} - \xi$*. If for some constant* $\epsilon_f \in (0, 1/2)$ *the reputation system* $\mathtt{Rep}$ *is* ϵ_f*-feasible for a static* $\mathtt{Rep}$*-bounded adversary* $\mathcal{A}$*, then for the set* $\mathcal{P}_{\mathtt{sel}}$ *of parties selected by* L *the following properties hold:*

1. *With overwhelming probability (in the security parameter* k*) :*
   - 1.a) $|\mathcal{P}_{\mathtt{sel}}| = \Theta(\log^{1+\epsilon} n)$*, and*
   - 1.b) *for some constant* $\epsilon_\delta > 0$ *adversary* $\mathcal{A}$ *corrupts at most an* $1/2 - \epsilon_\delta$ *fraction of the parties in* $\mathcal{P}_{\mathtt{sel}}$
2. *Additionally, if every set* $\hat{\mathcal{P}}_i$ *has at least* $\gamma \cdot \log^{1+\epsilon} n$ *parties for some* $\gamma > 1$*, then the lottery is c-fair.*

The complete proof can be found in Appendix B.2. In the following we included a sketch of the main proof ideas.

Idea. The proof considers two Cases.

In Case 1 the output $\mathcal{P}_{\mathtt{sel}}$ is selected by means of invocation of algorithm $\mathtt{A}_{\max}$. This is the simplest case since Lemma 1 ensures that if the reputation system is ϵ_f-feasible, then a fraction $1/2 + \epsilon_f$ of the parties in $\mathcal{P}_{\mathtt{sel}}$ will be honest except with negligible probability. Note that in this case, the number of selected parties is $|\mathcal{P}_{\mathtt{sel}}| = \log^{1+\epsilon} n$ with probability 1.

In Case 2, The lottery is never reset. Proving this case is the bulk of the proof.

First, in Lemma 3 using a combination of Chernoff bounds we prove that the number of parties from $\hat{\mathcal{P}}_i$ selected in the lottery is concentrated around the (expected) value:

$$x_i := Exp(X_i) = \alpha_i \cdot \sum_{j=i}^{m} \frac{\ell_j}{\sum_{q=1}^{j} \alpha_j}$$

[11] In the random oracle model, r can be any unique nonce; however, for the epoch-resettable-adversary extension of our lottery we will need r to be a sufficiently fresh random value. Although most of our analysis here is in the static setting, we will still have r be such a random value to ensure compatibility with dynamic reputation.

$\mathsf{L}(\hat{\mathcal{P}}, \mathtt{Rep}, (c_1, \ldots, c_m = 1, c_{m+1} = 0), \delta, \epsilon, r)$

1. Divide the reputation parties into m tiers $\hat{\mathcal{P}}_1, \ldots, \hat{\mathcal{P}}_m$ as follows,[a]: Set $\hat{\mathcal{P}}_1$ to the set of reputation parties with reputation $\mathtt{Rep}_j \in (0, \frac{1}{m} + \delta]$, and for $i = 2 \ldots, m$, define $\hat{\mathcal{P}}_i$ is the set of parties in $\hat{\mathcal{P}}$ with reputation $\mathtt{Rep}_j \in (\frac{i-1}{m} + \delta, \frac{i}{m} + \delta]$.
2. Initialize $\bar{\mathcal{P}}_i := \emptyset$ for each $i \in [m]$.
3. For each $i = 1, \ldots, m$: $\alpha_i := |\hat{\mathcal{P}}_i|$.
4. For each $i = 1, \ldots, m$:
   (a) Let
   $$\ell_i = \frac{\sum_{j=1}^{i} \alpha_j}{\sum_{j=1}^{m} \prod_{q=1}^{j} c_j} \frac{\alpha_{i+1} \prod_{j=0}^{m-i} c_{m-j} - \alpha_i \prod_{j=0}^{m-i-1} c_{m-j}}{\alpha_{i+1}\alpha_i} \log^{1+\epsilon} n$$
   (b) If $|\cup_{j=1}^{i} \hat{\mathcal{P}}_j| \geq \ell_i$:
      i. Invoke $\mathsf{Rand}(\cup_{j=1}^{i}\hat{\mathcal{P}}_j, \lceil \ell_i \rceil; (r||i))$ to choose a set $\mathcal{Q}_i$ of parties uniformly at random from $\cup_{j=1}^{i}\hat{\mathcal{P}}_j$.
      ii. For each $j \in [i]$ compute $\mathcal{Q}_{i,j}^{\text{col}} := \mathcal{Q}_i \cap \bar{\mathcal{P}}_j$ and update $\bar{\mathcal{P}}_j := \bar{\mathcal{P}}_j \cup (\mathcal{Q}_i \cap \hat{\mathcal{P}}_j)$.
      iii. For each $j \in [i]$ if $\mathcal{Q}_{i,j}^{\text{col}} \neq \emptyset$, then
         A. if $|\hat{\mathcal{P}}_j \setminus \bar{\mathcal{P}}_j| < |\mathcal{Q}_{i,j}^{\text{col}}|$ then reset the lottery and select $\mathcal{P}_{\mathtt{sel}}$ as the output of $\mathtt{A}_{\max}$.
         B. Else
            - Invoke $\mathsf{RandSet}(\hat{\mathcal{P}}_j \setminus \bar{\mathcal{P}}_j, |\mathcal{Q}_{i,j}^{\text{col}}|; (r||i||j))$ to choose a set $\mathcal{Q}_{i,j}^{+}$;
            - For each $j \in [i]$ update $\bar{\mathcal{P}}_j := \bar{\mathcal{P}}_j \cup \mathcal{Q}_{i,j}^{+}$.
         C. Set $\mathcal{Q}_i^{+} := \cup_{j=1}^{m} \mathcal{Q}_{i,j}^{+}$
   (c) Else (i.e., $|\cup_{j=1}^{i} \hat{\mathcal{P}}_j| < \ell_i$): Reset the lottery and select (and output) $\mathcal{P}_{\mathtt{sel}}$ as the output of $\mathtt{A}_{\max}$ for choosing $\log^{1+\epsilon} n$.
5. If the lottery was not reset in any of the above steps, then set $\mathcal{P}_{\mathtt{sel}} := \cup_{j=1}^{m} \bar{\mathcal{P}}_j (= \cup_{i=1}^{m} (Q_i \cup Q_i^{+}))$ and output $\mathcal{P}_{\mathtt{sel}}$.

[a] Where δ can be an arbitrary small constant.

Figure 1: c-fair reputation-based lottery for m=O(1) tiers

From this we deduce that the x_i's and the ℓ_j's satisfy the following system of linear equations:

$$(x_1, \ldots, x_m)^T = B \cdot (\ell_1, \ldots, \ell_m)^T \tag{1}$$

Where B is an $m \times m$ matrix with the (i, j) position being

$$B_{i,j} = \begin{cases} \frac{\alpha_i}{\sum_{q=1}^{j} \alpha_j}, \text{ if } i \geq j \\ 0, \text{ otherwise} \end{cases}$$

The above system of m equations has $2m$ unknowns. To solve it we add the following m equations which follow from the desired reputation fairness:

- For each $i \in [m-1]$:
$$x_{i+1} := c_i \cdot x_i \tag{2}$$

•

$$\sum_{i=1}^{m} x_i = \log^{1+\epsilon} k \tag{3}$$

This yields 2m linear equations. By solving the above system of equations we can compute:

$$\ell_i = \frac{\sum_{j=1}^{i} \alpha_j}{\sum_{j=1}^{m} \prod_{q=1}^{j} c_j} \frac{\alpha_{i+1} \prod_{j=0}^{m-i} c_{m-j} - \alpha_i \prod_{j=0}^{m-i-1} c_{m-j}}{\alpha_{i+1}\alpha_i} \log^{1+\epsilon} n$$

This already explains some of the mystery around the complicated choice of the ℓ_i's in the protocol.

Next we observe that for each $j \in [m] : \sum_{i=1}^{m} B_{i,j} = 1$ which implies that

$$\sum_{j=1}^{m} \ell_j = \sum_{i=1}^{m} x_i \overset{Eq.\ 3}{=} \log^{1+\epsilon} k \tag{4}$$

Because in each round we choose parties whose number is from a distribution centred around ℓ_i, the above implies that the sum of the parties we sample is centred around $\sum_{j=1}^{m} \ell_j = \log^{1+\epsilon} k$ which proves property 1.a).

Property 1.b) is then proved by a delicate counting of the parties which are corrupted, using Chernoff bounds for bounding the number of corrupted parties selected by Rand (which selects with replacement) and Hoeffding's inequality for bounding the number of parties selected by RandSet (which selects without replacement). The core idea of the argument is that because the reputation in different tiers reduces in a linear manner but the representation to the output of the lottery reduces in an exponential manner, even if we give to the adversary all the parties selected in the lowest half reputation tiers, still the upper half will have a very strong super-majority to compensate so that overall the majority is honest.

Finally, the c-fairness properties are argued as follows:

–The c-Representation Fairness follows directly from Lemma 3 and Equation 2.

–The non-discrimination property follows from the fact that our lottery picks each party in every $\hat{\mathcal{P}}_i$ with exactly the same probability as any other party.

–The c-Selection Fairness is proved by using the fact that the non-discrimination property mandates that each party in $\mathcal{P}_{\mathtt{sel}} \cap \hat{\mathcal{P}}_i$ is selected with probability $p_i = \frac{|\mathcal{P}_{\mathtt{sel}} \cap \hat{\mathcal{P}}_i|}{|\hat{\mathcal{P}}_i|}$. By using our counting of the sets' cardinalities computed above we can show that for any constant $c' < c$: $\frac{p_i}{p_{i+1}} \geq c'$.

□

5 From a PoR-Fair Lottery to a Blockchain

We next describe how a PoR-fair lottery can be used to obtain a PoR-fair blockchain. The idea is as follows: Time will be divided in epochs, where each epoch consists of a fixed number of rounds, aka *slots*.

The ground truth of the blockchain is recorded on the genesis block which includes the (initial) set of reputation parties, their public keys and the corresponding reputation vector. For simplicity, we focus our attention to the case of static reputation, i.e., the set of reputation parties is fixed and

so are their individual reputations. Appendix D.5 discusses an approach to a reputation-update mechanism; a formal treatment of such a mechanism is left as future work.

5.1 The Consensus Mechanism

In the following, as discussed in the model, we assume for simplicity that at the beginning of each epoch the parties receive from a beacon (cf. Section 3.3) a sufficient amount of fresh independent randomness to run the lotteries associated with the current epoch of our protocol. Using standard techniques from the PoS literature we can ensure that security of our construction is preserved even under imperfect randomness (cf. Section D.3). We assume a canonical way of validating transactions submitted in the same round, e.g., if two received transactions which have not-yet been inserted into a block would contradict each other (e.g., correspond to double-spending), a default rule of ignoring both can be adopted. We will abstract this by means of a predicate Validate, that takes as input a sequence T (of transactions) along with a current vector T_H of transaction history—composed by concatenating the transactions in the blockchain, and outputs a subset $T' \subseteq T$ such that $T_H||T'$ is a valid sequence of transactions.

The idea of the protocol is as follows: For each slot, a random small (i.e., polylogarithmically big) slot-committee $\mathcal{C}_{\text{BA}}$ is chosen using our above lottery—recall that the lottery will guarantee that the majority in $\mathcal{C}_{\text{BA}}$ is honest and therefore it can run Byzantine agreement protocols (Consensus and Broadcast). Note that whenever in our protocol we say that a party P broadcasts a message, we mean that a Byzantine broadcast protocol is executed with P as sender; to avoid confusion we will signify this by saying that that the message is broadcast by means of protocol Broadcast. For clarity in our description we will use a deterministic broadcast protocol for Broadcast, e.g., the Dolev-Strong broadcast protocol [DS83] for which we know the exact number of rounds. However, using the techniques from [CCGZ16, CCGZ17], we can replace the round-expensive Dolev-Strong broadcast protocol by an randomized, expected-constant round broadcast protocol for honest majorities, e.g., [KK06] in an almost blackbox manner. We stress that only parties in this committee will send messages in this respective slot, and as the identities of the committee can be computed by everyone, the parties never need to process messages that do not originate from the current slot committee members.[12] Out of this $\mathcal{C}_{\text{BA}}$ we will choose randomly a smaller (constant size) committee $\mathcal{C}_{\text{BC}}$; this committee will get to propose transactions to go to the currently formed block—the block will include the union of all the proposed transactions.[13]

Once the committee is chosen, the protocol proceeds as follows: Every party in $\mathcal{C}_{\text{BC}}$, in parallel, acts as a sender in an invocation of protocol Broadcast to broadcast his set of known transactions—aka its local transaction pool—to the set $\mathcal{C}_{\text{BA}}$ (once the broadcast protocol terminates, the respective sender removes the broadcasted transactions from its transaction pool). Since the majority in $\mathcal{C}_{\text{BA}}$ is honest, the broadcast will be successful, i.e., every party will output the same set of transactions, and all transactions broadcasted by any honest party in $\mathcal{C}_{\text{BC}}$ will be included in this set. In order to get convincing evidence that this is the right set, we will have every party in $\mathcal{C}_{\text{BA}}$ sign it. At this point we can forward it to the whole set of nodes and everyone can adopt the set which is backed by a majority of signatures from $\mathcal{C}_{\text{BA}}$—since the majority in $\mathcal{C}_{\text{BA}}$ is honest and honest parties agree on the set of transactions, only this set might be accepted.

However, as, in reality, some reputation parties will have public identities and IP addresses and can be reached by direct communication, we will use the following method for exploiting this fact

[12] We can make sure that all messages from non-committee members are ignored by having committee members attach standard size-efficient signatures.

[13] As usually, we assume in the analysis unlimited-size blocks; for dealing with bounded size blocks we will allocate a transaction budget to each of the parties in $\mathcal{C}_{\text{BA}}$.

to optimize communication: Rather than flood the entire set $\mathcal{P}$, the parties in $\mathcal{C}_{\text{BA}}$ will return their signed transaction sets to the parties in $\mathcal{C}_{\text{BC}}$, whenever possible over a direct channel. Then $\mathcal{C}_{\text{BC}}$—which, recall, has only a tiny (constant) number of members—will be responsible for multicasting it to the whole set $\mathcal{P}$. Note that this does not compromise consensus—in other words will not create forks—as (1) only transaction-sets with majority (from $\mathcal{C}_{\text{BA}}$ parties) support might make it into a block, and (2) from the properties of the multicast network, any honest party in $\mathcal{P}$ that accepts a new block will relay it and hence the block will reach all other honest parties (and for the same reason of honest majority support, this relay will convince any other honest receiver). Furthermore, unlike what one might initially think, having only $\mathcal{C}_{\text{BC}}$ players multicast blocks to the full node set does not affect transaction liveness[14] even though $\mathcal{C}_{\text{BC}}$ is not large enough to guarantee presence of an honest parties with overwhelming probability. Indeed, if there is an honest party in $\mathcal{C}_{\text{BC}}$ then this party will propagate the $\mathcal{C}_{\text{BA}}$-majority-backed transaction-set/block which will be adopted; otherwise, i.e., if all parties in $\mathcal{C}_{\text{BC}}$ are corrupted, then the worst that can happen is that they do not propagate a block for that round. As there is more than one party in $\mathcal{C}_{\text{BC}}$ this will yield a higher transaction liveness than existing solutions (see Section 1.2 for a discussions on the liveness guarantees of our protocol).

The formal description of the protocol for announcing and agreeing on the next block can be found in Figure 2. Recall that we assume that the parties maintain a set of received transactions (their local transaction pool) which is periodically, locally validated by means of the predicate $\mathsf{Validate}(\cdot)$ and updated whenever a new block is added on the chain by removing conflicting transactions. Recall that all parties are assumed to be available when selected in a committee. In Appendix D we discuss how reputation can be used to inceltivize availability. The proof of the following theorem can also be found in Appendix B.3.

Theorem 3. *Let* $\mathtt{Rep}$ *be a reputation system,* ϵ *and* δ *be small positive constants, and* L *denote our sampling algorithm (lottery) discussed in the previous section used for choosing* $\mathcal{C}_{BA}$ *according to* $\mathtt{Rep}$*. If for a static adversary* $\mathcal{A}$*,* $\mathtt{Rep}$ *is* ϵ*-feasible, and all parties in* $\hat{\mathcal{P}}_\rho$ *participate in slot* ρ *then in protocol* $\mathsf{BlockAnnounce}$*, every node in* $\mathcal{P}$ *adds the same block on his local blockchain for slot* ρ*. Moreover, this block will include the union of all transactions known to parties in* $\mathcal{C}_{BC}$ *at the beginning of round* ρ*.*

Using Protocol $\mathsf{BlockAnnounce}$ it is straightforward to build a blockchain: In each round parties collect all still valid transactions they know and exectute $\mathsf{BlockAnnounce}$. For completeness we also provide a description of our reputation-based blockchain protocol (see Figure 3). Its security follows directly from the above theorem. We remark that although the above theorem is proven for static reputations and adversaries, its proof can be extended, using standard techniques in the blockchain literature (cf. Section D.3), to dynamic reputation systems with epoch-resettable adversaries. This extension will also extend the corresponding PoR-based blockchain protocol to that setting.

Remark 1 (Joining parties)**.** *The static reputation assumption makes it easy to join the protocol as a user: In order for a user (who knows the current value of the clock) to read the blockchain he simply needs to query any of the parties for receiving all* ρ *blocks, where* ρ *is the current round, and check that all blocks are valid, i.e., they are properly signed by a majority of parties in a committee that is the result of running the lottery on the previous blocks. Extending this to dynamic reputation requires the use of key-evolving signatures and also requires all the reputation updates to be recorded on the blockchain before taking effect. A complete treatment of that case is left as a future research direction.*

[14]Informally, transaction liveness [GKL15] requires that transactions of honest parties make it into an adopted block within a short time interval.

$\mathsf{BlockAnnounce}(\hat{\mathcal{P}}, \mathcal{P}, \mathtt{Rep}, B_{\rho-1}, \rho, \delta, \epsilon, L = O(1))$:
Let $\hat{\mathcal{P}}_\rho$ denote the reputation parties in Slot ρ.

1. Extract the concatenation of randomness $r_{\rho-1}$ from the previous block.
2. Using the above randomness $r_{\rho-1}$,[a] each party in $\mathcal{P}$ locally runs the reputation-fair lottery $\mathsf{L}(\hat{\mathcal{P}}, \mathtt{Rep}, (c_1, \ldots, c_m + 1), \delta, \epsilon, \mathsf{h}(r_{\rho-1}||0))$, where the c_js are as in Theorem 2, to sample a set $\mathcal{C}_{\mathrm{BA}} \subset \hat{\mathcal{P}}_\rho$ (of size $polylog(n)$); out of this set, the parties choose a random subset $\mathcal{C}_{\mathrm{BC}}$ of constant size $L = O(1)$ by invoking $\mathsf{RandSet}(\mathcal{C}_{\mathrm{BA}}, L; \mathsf{h}(r_{\rho-1}||1))$.
3. Each party $\hat{P}_i \in \mathcal{C}_{\mathrm{BC}}$ acts as sender in an invocation of $\mathsf{Broadcast}$ with receivers the parties in $\mathcal{C}_{\mathrm{BA}}$ and input $\hat{P}_i$'s current transaction pool T_i along with a random nonce r_i; $\hat{P}_i$ removes the broadcasted transactions from its local transaction pool.
4. All parties in $\mathcal{C}_{\mathrm{BA}}$ compute $Y = (\hat{T}, r)$, where $\hat{T} = \mathsf{Validate}(T_H, T)$ for $T = \cup_{p_i \in \mathcal{C}_{\mathrm{BC}}} T_i$ and r is the concatenation of all broadcasted r_i's. If some party $\hat{P}_j \in \mathcal{C}_{\mathrm{BA}}$ did not broadcast a valid message in the previous round of the protocol, then all parties in $\mathcal{C}_{\mathrm{BA}}$ set $T_j = \{(abort, j)\}$ and $r_j = 0$.
5. Every $\hat{P}_j \in \mathcal{C}_{\mathrm{BA}}$ signs $\mathsf{h}(Y, h = \mathsf{h}(B_{\rho-1}), \rho)$, where $B_{\rho-1}$ is a hash pointer to the block of the previous round and sends it to every party in $\mathcal{C}_{\mathrm{BC}}$.
6. Each $\hat{P}_i \in \mathcal{C}_{\mathrm{BC}}$: If for some (Y, h, ρ), where ρ is the current slot and $h = \mathsf{h}(B_{\rho-1})$ is a valid hash pointer to the previous block, $\hat{P}_i$ receives at least $|\mathcal{C}_{\mathrm{BA}}|/2$ signatures from parties in $\mathcal{C}_{\mathrm{BA}}$ on (Y, h), then $\hat{P}_i$ multicasts (Y, h, ρ) along with all the corresponding signatures to all nodes in $\mathcal{P}$ and resets his local transaction-pool set to $\emptyset$.
7. Each $P_i \in \mathcal{P}$: Upon receiving any (Y, h, ρ) along with signatures on it from at least $|\mathcal{C}_{\mathrm{BA}}|/2$ parties from $\mathcal{C}_{\mathrm{BA}}$, create a block consisting of (Y, h, ρ) and the related signatures and add this block to the blockchain as the current slot's block and mark the current slot as completed.

[a] In the version of the protocol using a beacon, the parties can simply use the current-round value of the beacon.

Figure 2: Block announcing protocol for Slot ρ

$\mathsf{RepBC}(\hat{\mathcal{P}}, \mathcal{P}, \mathtt{Rep}, B_{\rho-1}, \rho, \delta, \epsilon, L = O(1))$
Every party receives the genesis block which includes: an encoding of the reputation system $\mathtt{Rep}$, and the public keys of all initial reputation parties. Let ρ_{BA} denote the number of (clock) rounds that each invocation of $\mathsf{BlockAnnounce}$ takes to complete.[a] Each slot in our protocol will be ρ_{BA} clock-rounds. Without loss of generalize, we assume that the protocol starts when the global clock round is 1.

- Extending the blockchain: In the ρth slot:
  1. Let $B_{\rho-1}$ denote the block adopted in the previous slot (where B_0 is the genesis block):
  2. The parties execute $\mathsf{BlockAnnounce}(\hat{\mathcal{P}}, \mathcal{P}, \mathtt{Rep}, B_{\rho-1}, \rho, \delta, \epsilon, L)$ to announce and agree on the ρ-th slot block.

[a] Recall that for simplicity we describe the protocol with deterministic Byzantine Agreement; using the techniques from [CCGZ16] we can extend our treatment to use randomized—and more round-efficient—protocols.

Figure 3: The Reputation-based Blockchain (Static Reputation)

6 A PoR/PoS-Hybrid Blockchain

A purely PoR-based blockchain protocol can be manipulated by malicious parties who behave honestly until they build reputation and then use their reputation to attack the system. To minimize the effect of such attacks on the protocol, we devise a hybrid-design approach: In parallel to the fast and sharp-finality PoR-based blockchain, we will run a slower, Proof-of-Stake-based blockchain (in short, PoS-blockchain) whose contents will be special transactions (see below) along with digests of the blocks of the PoR-blockchain. This *back-up PoS-blockchain* will ensure that PoR cannot be manipulated as above by exploiting an inaccurate reputation system, as long as the majority of the stake is in honest hands. In addition to offering a fallback guarantee, the PoS-based blockchain can be used in a bootstrapping phase to establish a common view on the reputation system (cf. Section D.5).

The guarantees that we will be receiving from such a hybrid blockchain design are the following: If the reputation system is accurate, i.e., the reputations reflect the actual probabilities that the parties behave honestly, then our protocol will be secure and achieve the claimed speed and finality. Additionally, if the reputation is inaccurate and, as a result, a fork is created, the parties will be able to detect this—and agree on this fact—within a small number of rounds as long as the majority of the stake in the system is in honest hands. Note that detection is a very important feature as it will ensure that no double spending can occur, which will not be observed very swiftly. Most importantly, our design will ensure that as long as the reputation system is good, the backup blockchain can neither yield false detection positives nor slow down the progress of the PoR-blockcain, even if the stake in the system has shifted to malicious parties (in which case the back-up PoS-blockchain might fork).

Here is how this can be achieved: In every round the reputation parties will be posting, as a transaction on the backup PoS-blockchain, the current slot's hash pointers. This way every party will be able to efficiently verify their local view by comparing their local hashes to the ones posted on the backup blockchain. If any honest party observes a discrepancy, then he can complain by posting the block in his local PoR-chain which is inconsistent with the pointer seen on the backup PoS-blockchain. We refer to this complaint as an *accusation* and to the party that posts it as an *accuser*. If the accuser fails to present a valid view (i.e., a block with sufficient signatures from the corresponding slot committee) then the accusation is considered invalid and the dispute is resolved in favor of the reputation party that had initially posted the disputed hash pointer, hereafter referred to as the *accused party*. Otherwise, the accused party will need to respond by posting (as a transaction on the backup PoS-blockchain) his own view of the disputed block. If this party fails, then his reputation will be reduced (see below) and the dispute is considered resolved in favor of the accuser. Otherwise, if both the accuser and the accused party post support on their claims, every party will be able to observe this fact and detect that the PoR-chain forked.The detailed specification of our PoR/PoS-hybrid protocol $\Pi^{BC}_{\text{PoR/PoS}}$ can be found in Figure 4, below.

Theorem 4 (see Appendix B.4 for a proof) states the achieved security, where we say that the reputation system is *accurate* if it reflects, within a tiny, negligible error the actual probabilities that the reputation parties are honest:

Theorem 4. *Let, ϵ, δ, c and L be as in Theorem 3. The following properties hold with overwhelming probability assuming that the reputation system is ϵ_f-feasible for some constant $\epsilon_f \in (0, 1)$: If the PoR-system is accurate, then protocol $\Pi^{BC}_{PoR/PoS}$ satisfies the following properties except with negligible probability in the presence of a static* `Rep`*-adversary:*

– *Safety (with Instant Finality): At the end of every slot, all parties have the same view of the blockchain.*

– *Block Liveness:* *A new block is added in every slot.*
– *Transaction Liveness:* *Assuming all honest reputation parties receive a transaction, this transaction will be added to the blockchain within ℓ slots except with probability $2^{-\ell|\mathcal{C}_{BC}|}$.*

Furthermore, even if the reputation system is faulty (e.g., inaccurate) but the majority of the stake is held by honest parties, then if safety is violated, $\Pi^{BC}_{PoR/PoS}$ will publicly detect it (i.e., agreement on this fact will be reached among all parties) within $2k$ blocks of the PoS-blockchain.

$\Pi^{BC}_{\text{PoR/PoS}}(\hat{\mathcal{P}}, \mathcal{P}, \texttt{Rep}, B_{\rho-1}, \rho, \delta, \epsilon, L = O(1))$
Every party receives the genesis block which includes: the reputation system $\texttt{Rep}$, the public keys of all initial reputation parties $\texttt{spk}_1, \ldots, \texttt{spk}_{|\hat{\mathcal{P}}|}$ (where each party $\hat{P}_i \in \hat{\mathcal{P}}$ knows the corresponding private key $\texttt{sprk}_i$), the public keys of all stake holders (users and reputation parties) in $\mathcal{P}$, i.e., $\texttt{pk}_1, \ldots, \texttt{pk}_{|\mathcal{P}|}$ (where each $P_i \in \mathcal{P}$ knows the corresponding private key $\texttt{rpk}_i$)[a], and the initial distribution of stake $S_1, \ldots, S_{|\mathcal{P}|}$. Let ρ_{BA} denote the number of (clock) rounds that each invocation of $\mathsf{BlockAnnounce}$ takes to complete.[b] Each slot in our protocol will be ρ_{BA} clock-rounds. Without loss of generalize, we assume that the protocol starts when the global clock round is 1. The parties execute the following processes in parallel:

- Extending the blockchain: In the ρth slot:
  1. Let $B_{\rho-1}$ denote the block adopted in the previous slot (where B_0 is the genesis block):
  2. The parties execute $\mathsf{BlockAnnounce}(\hat{\mathcal{P}}, \mathcal{P}, \texttt{Rep}, B_{\rho-1}, \rho, \delta, \epsilon, L)$ to announce and agree on the ρ-th slot block.
- Using the PoS blockchain as fallback: Invoke the PoS-blockchain where in each PoS-slot[c]:
  - The current PoS-slot leader P (who is in charge of posting the block for the current PoS-slot) creates a block as follows: If there are any valid complains, (see below) they are added to the block. Other than such complains, P includes to his block his view of the latest valid block on the PoR-blockchain without the actual transactions, i.e., if B_{ρ_L} is the latest block that P_i sees—where B_{ρ_L} includes $\mathsf{h}(Y, h = \mathsf{h}(B_{\rho_L-1}), \rho_L)$ as specified in $\mathsf{BlockAnnounce}$, then P_i includes in his PoS-block: $h = \mathsf{h}(Y, h = \mathsf{h}(B_{\rho_L-1}), \rho_L)$, the committee $\mathcal{C}_{\text{BA}}$ of the block B_{ρ_L}, and at least $\lceil|\mathcal{C}_{\text{BA}}|\rceil$ signatures from parties in $\mathcal{C}_{\text{BA}}$ on h.
  - Every party $P_i \in \mathcal{P}$ (independently of whether or not they are slot leaders): If P_i observes a view of a PoR-block B_j included in the PoS-blockchain which is different than his own view of that block (as extracted from the PoR-blockchain) then create a complaint-transaction for the PoS-blockchain transferring coins back to its own wallet which includes the conflicted PoS-block slot number, and his local view of block B_j (along with the associated signatures).
  - Every party $P_j \in \mathcal{P}$, if there is a valid complaint by any party on the PoS chain, then output $\texttt{Complaint}(q_1, q_2)$, where q_1 and q_2 are the PoS-blocks containing the disputed view of the PoR-blockchain

[a] Note that for simplicity we will separate the staking keys from the reputation keys. In reality, a reputation party can use its staking key as its reputation key.

[b] Recall that for simplicity we describe the protocol with deterministic Byzantine Agreement; using the techniques from [CCGZ16] we can extend our treatment to use randomized—and more round-efficient—protocols.

[c] Note that the slot in the PoS chain might be different than the slot in the PoR-chain. To make this distinction explicit we will refer to the former as a PoS-slot and to the latter as a PoR-slot

Figure 4: The PoR/PoS-Hybrid Blockchain Protocol (Static Reputation)

Remark 2 (Detection vs Recovery)**.** Our above hybrid design ensures that if any of the two independent assumptions—i.e., quality/accuracy of the reputation system and honest majority

of stake—holds, then no fork will settle on the distributed ledger—i.e., any fork will be quickly detected. This will be guaranteed by our protocol. This property is sufficient to make sure that the system does not preserve deep forks, which can lead to double spending. We remark, however, one can add support for recovery as follows: If malfunction of the PoR-blockchain is confirmed, then, as long as the majority of the stake is in honest hands, the parties will be able to reapply the bootstrapping mechanism (or other external reputation-system repairing mechanisms) discussed in Section D.5 to restart the PoR-blockchain from the point before the failure. The exact mechanism of this process and its rigorous analysis is interesting future research.

Acknowledgements. The authors would like to thank Yehuda Afek for useful discussions and his valuable feedback.

References

[AHR93] Hagit Attiya, Amir Herzberg, and Sergio Rajsbaum. Optimal clock synchronization under different delay assumptions (preliminary version). In Jim Anderson and Sam Toueg, editors, *12th ACM PODC*, pages 109–120. ACM, August 1993.

[ALZ13] Gilad Asharov, Yehuda Lindell, and Hila Zarosim. Fair and efficient secure multiparty computation with reputation systems. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 201–220. Springer, Heidelberg, December 2013.

[BFK17] Alex Biryukov, Daniel Feher, and Dmitry Khovratovich. Guru: Universal reputation module for distributed consensus protocols. Cryptology ePrint Archive, Report 2017/671, 2017. `http://eprint.iacr.org/2017/671`.

[BGK+18] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros genesis: Composable proof-of-stake blockchains with dynamic availability. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 913–930. ACM Press, October 2018.

[BGK+19] Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Ouroboros chronos: Permissionless clock synchronization via proof-of-stake. Cryptology ePrint Archive, Report 2019/838, 2019. `https://eprint.iacr.org/2019/838`.

[BHMN17] Iddo Bentov, Pavel Hubáček, Tal Moran, and Asaf Nadler. Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies. *IACR Cryptology ePrint Archive*, 2017:300, 2017.

[BM99] Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 431–448. Springer, Heidelberg, August 1999.

[BMTZ17] Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 324–356. Springer, Heidelberg, August 2017.

[But13] Vitalik Buterin. A next-generation smart contract and decentralized application platform, 2013. https://github.com/ethereum/wiki/wiki/White-Paper.

[CCGZ16] Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Probabilistic termination and composability of cryptographic protocols. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 240–269. Springer, Heidelberg, August 2016.

[CCGZ17] Ran Cohen, Sandro Coretti, Juan A. Garay, and Vassilis Zikas. Round-preserving parallel composition of probabilistic-termination cryptographic protocols. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *ICALP 2017*, volume 80 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl, July 2017.

[Cho07] Sherman S. M. Chow. Running on Karma - P2P reputation and currency systems. In Feng Bao, San Ling, Tatsuaki Okamoto, Huaxiong Wang, and Chaoping Xing, editors, *CANS 07*, volume 4856 of *LNCS*, pages 146–158. Springer, Heidelberg, December 2007.

[DGKR18] Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Heidelberg, April / May 2018.

[DHS84] Danny Dolev, Joseph Y. Halpern, and H. Raymond Strong. On the possibility and impossibility of achieving clock synchronization. In *16th ACM STOC*, pages 504–511. ACM Press, 1984.

[DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.

[DW93] Shlomi Dolev and Jennifer L. Welch. Wait-free clock synchronization (extended abstract). In Jim Anderson and Sam Toueg, editors, *12th ACM PODC*, pages 97–108. ACM, August 1993.

[DW95] Shlomi Dolev and Jennifer L. Welch. Self-stabilizing clock synchronization in the presence of byzantine faults (abstract). In James H. Anderson, editor, *14th ACM PODC*, page 256. ACM, August 1995.

[FM85] Paul Feldman and Silvio Micali. Byzantine agreement in constant expected time (and trusting no one). In *26th FOCS*, pages 267–276. IEEE Computer Society Press, October 1985.

[FM88] Paul Feldman and Silvio Micali. Optimal algorithms for byzantine agreement. In *20th ACM STOC*, pages 148–161. ACM Press, May 1988.

[GHM+17] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. Cryptology ePrint Archive, Report 2017/454, 2017. http://eprint.iacr.org/2017/454.

[GIOZ17] Juan A. Garay, Yuval Ishai, Rafail Ostrovsky, and Vassilis Zikas. The price of low communication in secure multi-party computation. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 420–446. Springer, Heidelberg, August 2017.

[GKKO07] Juan A. Garay, Jonathan Katz, Chiu-Yuen Koo, and Rafail Ostrovsky. Round complexity of authenticated broadcast with a dishonest majority. In *48th FOCS*, pages 658–668. IEEE Computer Society Press, October 2007.

[GKL15] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 281–310. Springer, Heidelberg, April 2015.

[GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, April 1988.

[GWDP18] Fangyu Gai, Baosheng Wang, Wenping Deng, and Wei Peng. Proof of reputation: A reputation-based consensus protocol for peer-to-peer network. In *DASFAA*, 2018.

[Hoe63] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[HSSD84] Joseph Y. Halpern, Barbara Simons, H. Raymond Strong, and Danny Dolev. Fault-tolerant clock synchronization. In Robert L. Probert, Nancy A. Lynch, and Nicola Santoro, editors, *3rd ACM PODC*, pages 89–102. ACM, August 1984.

[IR01] Gene Itkis and Leonid Reyzin. Forward-secure signatures with optimal signing and verifying. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 332–354. Springer, Heidelberg, August 2001.

[KK06] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 445–462. Springer, Heidelberg, August 2006.

[KRDO17] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 357–388. Springer, Heidelberg, August 2017.

[KZZ16] Aggelos Kiayias, Hong-Sheng Zhou, and Vassilis Zikas. Fair and robust multi-party computation using a global transaction ledger. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 705–734. Springer, Heidelberg, May 2016.

[LLW08] Christoph Lenzen, Thomas Locher, and Roger Wattenhofer. Clock synchronization with bounded global and local skew. In *49th FOCS*, pages 509–518. IEEE Computer Society Press, October 2008.

[LMS84] Leslie Lamport and P. M. Melliar-Smith. Byzantine clock synchronization. In Robert L. Probert, Nancy A. Lynch, and Nicola Santoro, editors, *3rd ACM PODC*, pages 68–74. ACM, August 1984.

[Mic17] Silvio Micali. Very simple and efficient byzantine agreement. In Christos H. Papadimitriou, editor, *ITCS 2017*, volume 4266, pages 6:1–6:1, 67, January 2017. LIPIcs.

[MMNT19] Bernardo Magri, Christian Matt, Jesper Buus Nielsen, and Daniel Tschudi. Afgjort - A semi-synchronous finality layer for blockchains. *IACR Cryptology ePrint Archive*, 2019:504, 2019.

[MXC+16] Andrew Miller, Yu Xia, Kyle Croman, Elaine Shi, and Dawn Song. The honey badger of BFT protocols. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 31–42. ACM Press, October 2016.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. `http://bitcoin.org/bitcoin.pdf`.

[OPS99] Rafail Ostrovsky and Boaz Patt-Shamir. Optimal and efficient clock synchronization under drifting clocks. In Brian A. Coan and Jennifer L. Welch, editors, *18th ACM PODC*, pages 3–12. ACM, May 1999.

[PS18] Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 3–33. Springer, Heidelberg, April / May 2018.

[PSs17] Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 643–673. Springer, Heidelberg, April / May 2017.

[ST85] T. K. Srikanth and Sam Toueg. Optimal clock synchronization. In Michael A. Malcolm and H. Raymond Strong, editors, *4th ACM PODC*, pages 71–86. ACM, August 1985.

[YKDE19] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo. Repucoin: Your reputation is your power. *IEEE Transactions on Computers*, 68(8):1225–1237, Aug 2019.

A Complementary Material to Section 4

$\mathsf{Rand}(\mathcal{P}, t; r)$

1. Initialize $\bar{\mathcal{P}} := \emptyset$.
2. For each $P_i \in \mathcal{P}$ with party-ID *pid*:
   (a) Compute $s_i = \mathsf{h}(pid, r)$.
   (b) If $T(s_i) \geq \log t$, then update $\bar{\mathcal{P}} := \bar{\mathcal{P}} \cup \{P_i\}$.
3. Output $\bar{\mathcal{P}}$.

Figure 5: A simple uniform sampler based on a hash function h

Lemma 2. *In the random oracle model and assuming $|\mathcal{P}| > \gamma t$ for some constant $1 < \gamma < 2$, and $t = \Omega(\log^{1+\epsilon} k)$ for some constant $\epsilon > 0$, the algorithm $\mathsf{Rand}(\mathcal{P}, t; r)$ has the following properties:*

1. *$Exp(|\bar{\mathcal{P}}|) = t$*
2. *For any constant $0 < \delta < 1$:*

$$\Pr\left[|(1-\delta)t \leq |\bar{\mathcal{P}}| \leq (1+\delta)t\right] \geq 1 - \mu(k),$$

   for some negligible function μ.
3. *For any $r' \neq r$, the output of $\mathsf{Rand}(\mathcal{P}, t; r')$ is distributed independently of the output of $\mathsf{Rand}(\mathcal{P}, t; r)$.*

Proof. For Properties 1 and 2, because by assumption h behaves as a random oracle, we know that for every P_i, $\Pr[P_i \in \bar{\mathcal{P}}] = \frac{t}{|\mathcal{P}|}$ independently of whether or not any other $P_i' \in \bar{\mathcal{P}}$. Hence, the random variable corresponding to $\bar{\mathcal{P}}$ is the sum of $|\mathcal{P}|$ independent Bernoulli trials with the above probability. Therefore, its expectation will be $Exp(|\bar{\mathcal{P}}|) = t$, and by a simple Chernoff bound, we have $\Pr\left[|(1-\delta)t \leq |\bar{\mathcal{P}}| \leq (1+\delta)t\right] \geq 1 - \mu(k)$. Property 3 also follows from the random oracle assumption which implies that the output of h on any inputs $x \neq x'$ is uniformly and independently distributed. □ □

$\mathsf{RandSet}(\mathcal{P}, t; r)$

1. Initialize $\bar{\mathcal{P}} := \emptyset$.
2. For each $P_i \in \mathcal{P}$ with party-ID *pid*[a]:
   (a) Compute $s_i = \mathsf{h}(pid, r)$.
   (b) Interpret $\mathsf{h}(pid, r)$ as a k-bit integer n_i.
3. Order the parties in $\mathcal{P}$ according to their corresponding n_i's.
4. Set $\bar{\mathcal{P}}$ to be the set of the first t parties in the above ordering.
5. Output $\bar{\mathcal{P}}$

[a] In our blockchain construction, *pid* will the P_i's public key.

Figure 6: A simple uniform sampler without replacements based on a hash function h

B Deferred Proofs

B.1 Proof of Theorem 1

Proof. We prove the two equations in the theorem separately. For brevity, we use the following notation: $\gamma = \max\left\{0, \frac{2\beta-\alpha}{\alpha+\beta}\right\}$

1. $\ell_I + \ell_{II} = \gamma\frac{y}{1+\gamma} + \frac{y}{1+\gamma} = y$
2. Let L_I denote the r.v. corresponding to the set of parties that are added on the committee in the first phase of the lottery and L_{II} the set added in the second and last phase. Denote by L the union, i.e., $L = L_I \cup L_{II}$. First we note that by definition:

$$\mathbb{E}(|X_i|) = \mathbb{E}(|L \cap \hat{\mathcal{P}}_i|)$$

Additionally, since in the second phase, parties from $\hat{\mathcal{P}}_2$ who were already chosen in the first phase are replaced by other (not-yet chosen) parties from $\hat{\mathcal{P}}_2$ we know that L_I and L_{II} are disjoint.

The following hold: As in the first phase of the lottery no party from $\hat{\mathcal{P}}_1$ is chosen an all parties are chosen from $\hat{\mathcal{P}}_2$:

$$\mathbb{E}(|L_I \cap \hat{\mathcal{P}}_1|) = 0 \tag{5}$$

and

$$\mathbb{E}(|L_I \cap \hat{\mathcal{P}}_2|) = \gamma\frac{y}{1+\gamma} \tag{6}$$

In the second phase, there are $\alpha+\beta$ parties to choose from in total, out of which α are in $\hat{\mathcal{P}}_2$ and β are in $\hat{\mathcal{P}}_1$. Since we chose $\frac{y}{1+\gamma}$ parties randomly we have:

$$\mathbb{E}(|L_{II} \cap \hat{\mathcal{P}}_1|) = \frac{\beta}{\alpha+\beta} \cdot \frac{y}{1+\gamma} \tag{7}$$

and

$$\mathbb{E}(|L_{II} \cap \hat{\mathcal{P}}_2|) = \frac{\alpha}{\alpha+\beta} \cdot \frac{y}{1+\gamma} \tag{8}$$

Using the above equations we can compute the expectations of X_1 and X_2:

$$\begin{aligned}
\mathbb{E}(|X_1|) &= \mathbb{E}(|L \cap \hat{\mathcal{P}}_1|) \\
&= \mathbb{E}(|L_I \cap \hat{\mathcal{P}}_1| + |L_{II} \cap \hat{\mathcal{P}}_1|) \\
&= \mathbb{E}(|L_I \cap \hat{\mathcal{P}}_1|) + \mathbb{E}(|L_{II} \cap \hat{\mathcal{P}}_1|) \\
&= 0 + \frac{\beta}{\alpha+\beta} \cdot \frac{y}{1+\gamma}
\end{aligned}$$

where the second equation holds because L_I and L_{II} are a partition of L and the third follows from the linearity of expectation.

Similarly,

$$\begin{aligned}\mathbb{E}(|X_2|) &= \mathbb{E}(|L \cap \hat{\mathcal{P}}_2|) \\ &= \mathbb{E}(|L_I \cap \hat{\mathcal{P}}_2| + |L_{II} \cap \hat{\mathcal{P}}_2|) \\ &= \mathbb{E}(|L_I \cap \hat{\mathcal{P}}_2|) + \mathbb{E}(|L_{II} \cap \hat{\mathcal{P}}_2|) \\ &= \gamma \frac{y}{1+\gamma} + \frac{\alpha}{\alpha+\beta} \cdot \frac{y}{1+\gamma} \\ &= \left(\gamma + \frac{\alpha}{\alpha+\beta}\right) \cdot \frac{y}{1+\gamma}\end{aligned}$$

To complete the proof we observe that by definition, γ is either 0 or $\frac{2\beta-\alpha}{\alpha+\beta} > 0$.

- When $\gamma = 0$:

$$\frac{\mathbb{E}(|X_2|)}{\mathbb{E}(|X_1|)} = \frac{\frac{\alpha y}{\alpha+\beta}}{\frac{\beta y}{\alpha+\beta}} = \frac{\alpha}{\beta}$$

  Since by definition $\gamma = 0$ only when $\alpha > 2\beta$ we have $\frac{\alpha}{\beta} > 2$ in this case hence $\mathbb{E}(|X_2|) = \max\{2, \frac{\alpha}{\beta}\}\,\mathbb{E}(|X_1|)$

- When $\gamma = \frac{2\beta-\alpha}{\alpha+\beta} > 0$:

$$\frac{\mathbb{E}(|X_2|)}{\mathbb{E}(|X_1|)} = \frac{\gamma + \frac{\alpha}{\alpha+\beta}}{\frac{\beta}{\alpha+\beta}} = \frac{\frac{2\beta-\alpha}{\alpha+\beta} + \frac{\alpha}{\alpha+\beta}}{\frac{\beta}{\alpha+\beta}} = 2 = \max\{2, \frac{\alpha}{\beta}\}$$

  where the last equation holds because $\gamma > 0$ occurs only when $\alpha \leq 2\beta$.

□

B.2 Proof of Theorem 2

Proof. Without loss of generality we will assume that all ℓ_i's above are integers, to avoid unnecessary rounding-related discussions. This assumption is without loss of generality as removing it might result in the samplers adding at most a constant number of parties on the sampled sets which will not change any of our asymptotic statement about the (relative) set sizes; furthermore, even if those additional parties are all corrupted, the asymptotic fraction of corrupted over honest parties will not change, since, as we prove, the sets are all super-constant with overwhelming probability and our corruption bound ensures that the adversary is far from the majority by an at least ϵ_δ fraction of the selected parties.

We consider two cases: Case 1: L resets, and Case 2: L does not reset.

In Case 1 the output $\mathcal{P}_{\mathtt{sel}}$ is selected by means of invocation of algorithm $\mathtt{A}_{\max}$. Hence, by Lemma 1, if the reputation system is ϵ_f-feasible, then a fraction $1/2 + \epsilon_f$ of the parties in $\mathcal{P}_{\mathtt{sel}}$ will be honest except with negligible probability. Note that in this case, the number of selected parties is $|\mathcal{P}_{\mathtt{sel}}| = \log^{1+\epsilon} n$ with probability 1.

In Case 2, i.e., conditioned on the lottery never being reset to its default, we prove the following lemmas:

For each $i \in [m]$: let X_i denote the random variable corresponding to the number of parties from $\hat{\mathcal{P}}_i$ selected in the lottery and let $x_i := Exp(X_i)$

Lemma 3. *For each $i \in [m]$:*

1.

$$x_i := Exp(X_i) = \alpha_i \cdot \sum_{j=i}^{m} \frac{\ell_j}{\sum_{q=1}^{j} \alpha_j}$$

2. For any constant $\lambda_i \in (0,1)$:

$$\Pr\left[|(1-\lambda_i)x_i \leq X_i \leq (1+\lambda_i)x_i\right] \geq 1 - \mu_i(k),$$

Proof. Let $\hat{\mathcal{P}}_i = \{\hat{P}_{i_1}, \ldots, \hat{P}_{i_{m_i}}\}$. For each $\hat{P}_{i_\rho} \in \hat{\mathcal{P}}_i$ and denote by $\chi_{i_\rho,j}$ the indicator random variable which is 1 if $\hat{P}_{i_\rho}$ is selected in the jth iteration of Step **4(b)i**, i.e., if $\hat{P}_{i_\rho} \in \mathcal{Q}_j$, and 0 otherwise. The j-th iteration of Step **4(b)i** chooses every party in $\cup_{q=1}^{j} \hat{\mathcal{P}}_q$ with probability $\frac{\ell_i}{\sum_{q=1}^{j} \alpha_q}$ independently of whether or not any other party in $\cup_{q=1}^{j} \hat{\mathcal{P}}_q$ is chosen. Indeed, observe that in the jth iteration of Step **4(b)i**, the lottery L samples always from the set $\cup_{q=1}^{j} \hat{\mathcal{P}}_q$ *with replacement* and conflicts are resolved later. Because, out of the $\sum_{q=1}^{j} \alpha_q$ parties in $\cup_{q=1}^{j} \hat{\mathcal{P}}_q$, α_i parties are from from $\hat{\mathcal{P}}_i$, $\Pr[\chi_{i_\rho,j} = 1] = \alpha_i \frac{\ell_i}{\sum_{q=1}^{j} \alpha_q}$ independent of whether or not any other party is chosen in $\mathcal{Q}_j$. Hence, each $\chi_{i_\rho,j}$ corresponds to a Bernoulli trial with the above success probability and by application of the Chernoff bound for the r.v. $X_{i,j} := \sum_{\rho=1}^{m_i} \chi_{i_\rho,j}$ corresponding to the number of parties from $\hat{\mathcal{P}}_i$ that are selected in Q_j:

$$x_{i,j} := Exp(X_{i,j}) = \alpha_i \cdot \frac{\ell_j}{\sum_{q=1}^{j} \alpha_j} \tag{9}$$

and for any $\lambda_{i,j} \in (0,1)$ and some negligible function $\mu_{i,j}$:

$$\Pr\left[|(1-\lambda_{i,j})x_{i,j} \leq X_{i,j} \leq (1+\lambda_{i,j})x_{i,j}\right] \geq 1 - \mu_{i,j}(k), \tag{10}$$

Next we observe that although some of the parties in $\mathcal{Q}_j \cap \hat{\mathcal{P}}_i$ selected in Step **4(b)i** might have been already selected and included in $\mathcal{P}_{\mathtt{sel}}$ (those are the parties in $\mathcal{Q}_{i,j}^{\mathrm{col}}$), by selecting exactly $|\mathcal{Q}_{i,j}^{\mathrm{col}}|$ parties *without replacement* from $\hat{\mathcal{P}}_i$, we ensure that the total number of parties selected in the jth iteration of Step 4(b) is exactly $X_{i,j}$.

To complete the proof we observe that $X_i = \sum_{j=1}^{m} X_{i,j}$, hence by the linearity of expectation Equation 9 implies that

$$x_i := Exp(X_i) = \alpha_i \cdot \sum_{j=i}^{m} \frac{\ell_j}{\sum_{q=1}^{j} \alpha_j}$$

and Equation 10 implies that for any $(\lambda_{i,1}, \ldots, \lambda_{i,m}) \in (0,1)^m$

$$\Pr\left[|(1-\sum_{j=1}^{m} \lambda_{i,j})x_{i,j} \leq \sum_{j=1}^{m} X_{i,j} \leq (1+\sum_{j=1}^{m} \lambda_{i,j})x_{i,j}\right] \geq 1 - \mu_i(k), \tag{11}$$

for some negligible function μ_i (recall that constant-term sums and products of constantly many negligible functions are also negligible). Hence, by setting $\lambda_{i,j} = \lambda_i/m$ (recall that $m = O(1)$ hence λ_i/m is a constant) we derive the second property of the lemma. □

Given the above lemma, it is easy to verify that the x_i's and the ℓ_j's satisfy the following system of linear equations:

$$(x_1, \ldots, x_m)^T = B \cdot (\ell_1, \ldots, \ell_m)^T \tag{12}$$

Where B is an $m \times m$ matrix with the (i, j) position being

$$B_{i,j} = \begin{cases} \frac{\alpha_i}{\sum_{q=1}^{j} \alpha_j}, \text{ if } i \geq j \\ 0, \text{ otherwise} \end{cases}$$

The above system of m equations has $2m$ unknowns. We add the following m equations:

- For each $i \in [m-1]$:

$$x_{i+1} := c_i \cdot x_i \tag{13}$$

-

$$\sum_{i=1}^{m} x_i = \log^{1+\epsilon} k \tag{14}$$

This yields 2m linear equations. By solving the above system of equations we can compute:

$$\ell_i = \frac{\sum_{j=1}^{i} \alpha_j}{\sum_{j=1}^{m} \prod_{q=1}^{j} c_j} \frac{\alpha_{i+1} \prod_{j=0}^{m-i} c_{m-j} - \alpha_i \prod_{j=0}^{m-i-1} c_{m-j}}{\alpha_{i+1}\alpha_i} \log^{1+\epsilon} n$$

We next observe that for each $j \in [m] : \sum_{i=1}^{m} B_{i,j} = 1$ which implies that

$$\sum_{j=1}^{m} \ell_j = \sum_{i=1}^{m} x_i \overset{Eq.\ 14}{=} \log^{1+\epsilon} k \tag{15}$$

It is also easy to verify that B is invertible, hence

$$(\ell_1, \ldots, \ell_m)^T = B^{-1}(x_1, \ldots, x_m)^T \tag{16}$$

We are now ready to prove properties $1.a)$ and $1.b)$ in the theorem. We start with Property $1.a)$:

Claim 1. *With overwhelming probability (in the security parameter k) : $|\mathcal{P}_{\mathtt{sel}}| = \Theta(\log^{1+\epsilon} n)$*

Proof. Let L_j denote the random variable corresponding to $|\mathcal{Q}_j|$, i.e., the total number of parties added to $\mathcal{P}_{\mathtt{sel}}$ in the jth iteration of Step 4. Lemma 2 implies that that for each $j \in [m]$:

$$Exp(|L_j|) = \ell_j \tag{17}$$

and for any constant $\zeta_j \in (0, 1)$ and some negligible function $\hat{\mu}_j$:

$$\Pr\left[|(1-\zeta_j)\ell_j \leq L_j \leq (1+\zeta_j)\ell_j\right] \geq 1 - \hat{\mu}_j(k), \tag{18}$$

Let $P_{\mathtt{sel}}$ denote the r.v. corresponding to the selected party set. By definition of the protocol: $P_{\mathtt{sel}} := \sum_{j=1}^{m} L_j$. Hence from the linearity of expectation and Equation 17 we have

$$Exp(|\mathcal{P}_{\mathtt{sel}}|) = \sum_{j=1}^{m} \ell_j \overset{Eq.15}{=} log^{1+\epsilon} k$$

Similarly, from Equation 18 we get

$$\Pr\left[|(1-\sum_{j=1}^{m}\zeta_j)\sum_{j=1}^{m}\ell_j \leq \sum_{j=1}^{m}L_j \leq (1+\sum_{j=1}^{m}\zeta_j)\sum_{j=1}^{m}\ell_j\right] \geq 1-\hat{\mu}(k), \tag{19}$$

for some negligible function $\hat{\mu}$, which, for any given ζ, by setting each $j \in [m] : \zeta_j = \zeta/m$, and setting $\hat{\mu}(\cdot) = \sum_{j=1}^{m}\hat{\mu}(\cdot)$ (which is also negligible when each $\hat{\mu}_j$ is negligible) and $\sum_{j=1}^{m}\ell_j = \log^{1+\epsilon} k$ derives

$$\Pr\left[|(1-\zeta)\log^{1+\epsilon} k \leq |P_{\mathtt{sel}}| \leq (1+\zeta)\log^{1+\epsilon} k\right] \geq 1-\hat{\mu}(k), \tag{20}$$

which implies that with overwhelming probability $|P_{\mathtt{sel}}| = \Theta(\log^{1+\epsilon} k)$.

□

We next proceed to Property 1.*b*):

Lemma 4. *With overwhelming probability (in the security parameter k) for some constant $\epsilon_\delta > 0$ adversary $\mathcal{A}$ corrupts at most an $1/2 - \epsilon_\delta$ fraction of the parties in $\mathcal{P}_{\mathtt{sel}}$.*

Proof. The reputation system $\mathtt{Rep}$ assigns to each party $\hat{\mathcal{P}}_i$ a reputation $\mathtt{R}_i \in [0,1]$ which corresponds to the probability that $\hat{\mathcal{P}}_i$ is honest. (Recall that we for this proof we consider static reputation systems.) By the independent reputations assumption, this probability is independent of whether or not any other party in $\hat{\mathcal{P}}$ becomes corrupted. This induces a probabilistic adversary $\mathcal{A}$ who corrupts each reputation party $\hat{\mathcal{P}}_i$ independently with probability $1 - \mathtt{R}_i$. We wish to prove that this adversary corrupts at most a $1/2 - \epsilon_\delta$ fraction of the parties in $\mathcal{P}_{\mathtt{sel}}$. We will prove this by considering an adversary $\mathcal{A}'$ which is stronger than $\mathcal{A}$, i.e, where $\Pr[\mathcal{A}'$ corrupts more than $1/2-\epsilon$ parties in $\mathcal{P}_{\mathtt{sel}}] \geq \Pr[\mathcal{A}$ corrupts more than $1/2-\epsilon$ parties in $\mathcal{P}_{\mathtt{sel}}]$, and proving that this adversary $\mathcal{A}'$ corrupts more than $1/2 - \epsilon$ parties with only negligible probability.

Here is how $\mathcal{A}'$ is defined: For each $\hat{\mathcal{P}}_i$ (recall that this includes parties with reputation between $(\frac{i-1}{m}+\delta, \frac{i}{m}+\delta]$), $\mathcal{A}'$ corrupts $\hat{\mathcal{P}}_i$ with probability $1-(\frac{i-1}{m}+\delta)$. Note that for each party $\hat{P} \in \hat{\mathcal{P}} : \Pr[\mathcal{A}'$ corrupts $\hat{P}] \geq \Pr[\mathcal{A}$ corrupts $\hat{P}]$ and this probabilities are independent of whether $\mathcal{A}'$ (resp. $\mathcal{A}$) corrupts any other party. This ensures the above property between $\mathcal{A}'$ and $\mathcal{A}$. Hence, it suffices to prove the lemma for $\mathcal{A}'$ which is what we do in the following.

Claim 2. *In the ith iteration of Step 4, let $C_{i,j}$ denote the random variable corresponding to the number of corrupted parties in $\mathcal{Q}_{i,j} := (\mathcal{Q}_i \cap \hat{\mathcal{P}}_j) \cup \mathcal{Q}^+_{i,j}$, i.e., the number of parties from $\hat{\mathcal{P}}_j$ that are newly added to $\mathcal{P}_{\mathtt{sel}}$ and are corrupted by $\mathcal{A}'$. Then for some negligible function μ for any constant $0 \leq \delta' < \frac{m-i}{m}$:*

$$\Pr[C_{i,j} < (\frac{m-i}{m} - \delta')|\mathcal{Q}_{i,j}|] > 1 - \mu(k).$$

Proof. We consider three cases: Case 1: $\mathcal{Q}^{\mathrm{col}}_{i,j} = o(|\mathcal{Q}_{i,j}|)$; Case 2: $\mathcal{Q}^{\mathrm{col}}_{i,j} = \omega(|\mathcal{Q}_{i,j}|)$; and Case 3: $\mathcal{Q}^{\mathrm{col}}_{i,j} = \Theta(|\mathcal{Q}_{i,j}|)$.

For Case 1: Since $\mathcal{Q}^{\mathrm{col}}_{i,j} = o(|\mathcal{Q}_{i,j}|)$, this implies that $|(\mathcal{Q}_i \cap \hat{\mathcal{P}}_j) \setminus \bar{\mathcal{P}}_j| = O(|\mathcal{Q}_i|)$. Hence Equation 18 implies that with overwhelming probability $|(\mathcal{Q}_i \cap \hat{\mathcal{P}}_j) \setminus \bar{\mathcal{P}}_j| = O(\log^{1+\epsilon} k)$. However, since each party in $\mathcal{Q}_i$ is chosen independently and the reputation is static, i.e., corruptions are defined before selecting the parties to join $\mathcal{Q}_i$, every party in $(\mathcal{Q}_i \cap \hat{\mathcal{P}}_j) \setminus \bar{\mathcal{P}}_j$ is corrupted by $\mathcal{A}'$ with probability $1-(\frac{i}{m}+\delta)$. But then an invocation of the Chernoff bound yields that with overwhelming probability, for any $\delta'' > 0$ the fraction of corrupted parties in $(\mathcal{Q}_i \cap \hat{\mathcal{P}}_j) \setminus \bar{\mathcal{P}}_j$ will be at most $\frac{m-i}{m} - \delta + \delta''$. Additionally since $\mathcal{Q}^{\mathrm{col}}_{i,j} = o(|\mathcal{Q}_{i,j}|)$, this implies that for any constant $\delta''' \in (0,1)$ and for sufficiently

large k $|\mathcal{Q}^{\text{col}}_{i,j}| < \delta'''|\mathcal{Q}_{i,j}|$. Hence, even if we allow the adversary $\mathcal{A}'$ to corrupt all parties in $\mathcal{Q}^{\text{col}}_{i,j}$, for $\delta + \delta''' - \delta'' < \epsilon_\delta$ we will have that with overwhelming probability the fraction of corrupted parties in $\mathcal{Q}_{i,j}$ will be at most $\frac{m-i}{m} - \epsilon_\delta$.

For Case 2: Since $\mathcal{Q}^{\text{col}}_{i,j} = \omega(|\mathcal{Q}_{i,j}|)$, this implies that $|\mathcal{Q}^{+}_{i,j}| = O(|Q_i|)$. Hence Equation 18 implies that with overwhelming probability $|\mathcal{Q}^{+}_{i,j}| = O(\log^{1+\epsilon} k)$. However, unlike the case above each party in $\mathcal{Q}^{+}_{i,j}$ is not chosen independently, but rather a set of parties is chosen randomly. This corresponds to sampling with replacement and we can no longe use the Chernoff bound. But since the reputation is static, i.e., corruptions are defined before selecting the above set, we can make direct use of Hoeffdings inequality [Hoe63] (see Appendix C), to prove that with overwhelming probability, the fraction of corrupted parties in $\mathcal{Q}^{+}_{i,j}$ will be at most $\frac{m-i}{m} - \delta + \delta''$. Additionally since $\mathcal{Q}^{\text{col}}_{i,j} = \omega(|\mathcal{Q}_{i,j}|)$, this implies that for any constant $\delta''' \in (0,1)$ and for sufficiently large k $|(\mathcal{Q}_i \cap \hat{\mathcal{P}}_j) \setminus \bar{\mathcal{P}}_j| < \delta'''|\mathcal{Q}_{i,j}|$. Hence, even if we allow the adversary $\mathcal{A}'$ to corrupt all parties in $(\mathcal{Q}_i \cap \hat{\mathcal{P}}_j) \setminus \bar{\mathcal{P}}_j$, for $\delta + \delta''' - \delta'' < \epsilon_\delta$ we again will have that with overwhelming probability the fraction of corrupted parties in $\mathcal{Q}_{i,j}$ will be at most $\frac{m-i}{m} - \epsilon_\delta$.

Finally in Case 3, $\mathcal{Q}^{\text{col}}_{i,j} = \Theta(|\mathcal{Q}_{i,j}|)$ impies that $|(\mathcal{Q}_i \cap \hat{\mathcal{P}}_j) \setminus \bar{\mathcal{P}}_j| = O(\log^{1+\epsilon} k)$ and $|\mathcal{Q}^{+}_{i,j}| = O(\log^{1+\epsilon} k)$. Hence by using both arguments from the above two cases we can prove that (1) with overwhelming probability the fraction of corrupted parties in $(\mathcal{Q}_i \cap \hat{\mathcal{P}}_j) \setminus \bar{\mathcal{P}}_j$ will be at most $(\frac{m-i}{m} - \epsilon_\delta/2)|(\mathcal{Q}_i \cap \hat{\mathcal{P}}_j) \setminus \bar{\mathcal{P}}_j|$ and (2) with overwhelming probability the fraction of corrupted parties in $\mathcal{Q}^{+}_{i,j}$ will be at most $(\frac{m-i}{m} - \epsilon_\delta/2)$ of the size of Q^{+}_{j}. Therefore by a union bound, the fraction of corrupted parties in $Q_{i,j}$ will be at most $\frac{m-i}{m} - \epsilon_\delta$.

□

Next, we observe by inspection of the protocol that $|\mathcal{Q}_{i,j}| = X_{i,j}$. Since from for any constant $\lambda_{i,j} > 0$, Equation 10 implies that

$$\Pr\left[X_{i,j} \leq (1 + \lambda_{i,j}) x_{i,j}\right] \geq 1 - \hat{\mu}_{i,j}(k),$$

for some negligible function $\hat{\mu}_{i,j}$, and i, m, and $\lambda_{i,j}$ are all constants, this implies that for any sufficiently small positive constant δ', and for some negligible function $\mu'_{i,j}$:

$$\Pr[C_{i,j} < (\frac{m-i}{m} - \delta') x_{i,j}] \geq 1 - \mu'_{i,j}(k). \tag{21}$$

But then from a union bound we can deduce that there exists a negligible function $\tilde{\mu}_i$ such that:

$$\Pr[\forall j : \; C_{i,j} < (\frac{m-i}{m} - \delta') x_{i,j}] \geq 1 - \tilde{\mu}_i(k). \tag{22}$$

which implies that

$$\Pr[\sum_{j=1}^{m} C_{i,j} < \sum_{j=1}^{m} (\frac{m-i}{m} - \delta') x_{i,j}] \geq 1 - \mu''_i(k), \tag{23}$$

for some negligible μ''_i. Denote by C_i the total number of corrupted parties from $\hat{\mathcal{P}}_i$ chosen in the lottery. By inspection of the protocol:

$$C_i = \sum_{j=1}^{m} C_{i,j}$$

Hence Equation 23 can be rewritten as follows

$$\Pr[C_i < (\frac{m-i}{m} - \delta')x_i] \geq 1 - \mu_i''(k) \tag{24}$$

Wlog assume that m is even (the case when m is odd is analogous):

For each $i \in \{2, \ldots, m/2-1\}$ the above implies that the majority of the parties in C_i is honest with overwhelming probability. Hence, but a union bound, the majority of the parties in $\cup_{i=2}^{m/2-1} Q_i$ is honest with overwhelming probability. To complete the proof it suffices to prove that a $(1/2-\epsilon'_\delta)$-fraction of the parties in $Q_1 \cup (\cup_{j=m/2}^{m} Q_j)$ is honest with overwhelming probability. To this direction we observe that from Equation 24, with overwhelming probability, $\Pr[C_1 < (\frac{1}{m} - \delta')x_1]$. By an easy calculation, it follows that for any δ' and any c such that $\sum_{i=m/2}^{m} \frac{1}{c^{i-1}} \leq \frac{m-2}{2m} - \delta'$ (which can be equivalently written as $\frac{1}{c^{m-1}} \leq \frac{m-2}{2m} - \delta'$) the total number of parties in $\cup_{i=m/2}^{m} Q_j$ is less than $(\frac{m-2}{2m} - \delta')x_1$. But if this is the case then then even if we allow the adversary to corrupt all parties in every Q_j (note that this is an even stronger adversary than $\mathcal{A}'$), still with overwhelming probability the total number $corr_1$ of corrupted parties in $Q_1 \cup (\cup_{i=m/2}^{m} Q_j)$ will be:

$$\begin{aligned} corr_1 &= C_1 + \frac{m-2}{2m}x_1 \\ &< (\frac{1}{m} + \frac{m-2}{2m} - \delta')x_1 \\ &= (1/2 - \delta')x_1 \end{aligned} \tag{25}$$

Hence with overwhelming probability, the fraction R of corrupted parties in $Q_1 \cup (\cup_{i=m/2}^{m} Q_j)$ will be

$$R < \frac{(1/2 - \delta')x_1}{x_1 + \sum_{i=m/2}^{m} x_i} < 1/2 - \delta' \tag{26}$$

□

We next argue the following which will ensure that under Condition 2 of the protocol, i.e., that every set $\hat{\mathcal{P}}_i$ has at least $\gamma \log^{1+\epsilon} n$ parties, with overwhelming probability the lotterty never resets and hence all the claims in this second case hold. This follows directly from the fact that the total number of parties selected in the lottery will be less than $\gamma \log^{1+\epsilon} n$ with overwhelming probability; hence, none of the invocation of Step 4 exceeds the size of the corresponding sets and therefore the lottery never resets.

To complete the proof we show the c-fairness property of L in this case.

The c-Representation Fairness follows directly from Lemma 3 and Equation 13.

The non-discrimination property follows from the fact that our lottery picks each party in every $\hat{\mathcal{P}}_i$ with exactly the same probability as any other party.

The c-Selection Fairness is proved as follows: Since by the non-discrimination property every party has the same probability of being picked, each party in each $\mathcal{P}_{\mathtt{sel}} \cap \hat{\mathcal{P}}_i$ is chosen with probability $p_i = \frac{|\mathcal{P}_{\mathtt{sel}} \cap \hat{\mathcal{P}}_i|}{|\hat{\mathcal{P}}_i|}$. However from Lemma 3 we know that with overwhelming probability, for any constant $\lambda_i \in (0,1)$:

$$(1-\lambda_i)x_i \leq |\mathcal{P}_{\mathtt{sel}} \cap \hat{\mathcal{P}}_i| \leq (1+\lambda_i)x_i$$

This means that with overwhelming probability, for all $i = 1, \ldots, m-1$:

$$\begin{aligned}\frac{p_i}{p_{i+1}} &\geq \frac{(1-\lambda_i)x_i}{(1+\lambda_{i+1})x_{i+1}} \cdot \frac{\alpha_{i+1}}{\alpha_i} \\ &= \frac{(1-\lambda_i)}{(1+\lambda_{i+1})} \cdot \frac{x_i}{x_{i+1}} \cdot \frac{\alpha_{i+1}}{\alpha_i} \\ &= \frac{(1-\lambda_i)}{(1+\lambda_{i+1})} \cdot c_i \cdot \frac{\alpha_{i+1}}{\alpha_i} \\ &\geq \frac{(1-\lambda_i)}{(1+\lambda_{i+1})} c\end{aligned} \tag{27}$$

Where the last inequality follows by the definition of c_i. For any constant $c' < c$, by choosing λ_i and λ_{i+1} such that $\frac{(1-\lambda_i)}{(1+\lambda_{i+1})} \geq c'/c$ we can ensure that $\frac{p_i}{p_{i+1}} \geq c'$.

□

B.3 Proof of Theorem 3

Proof. Assume that every party has received the same genesis block. This block includes the identifiers (public keys) of all parties currently in $\hat{\mathcal{P}}$ and their reputations (recall that for this proof, we assume static reputations). Since $\mathcal{C}_{\text{BA}}$ is selected by means of L, and the lottery is ϵ-feasible, Theorem 2 ensures that the majority of the parties in $\mathcal{C}_{\text{BA}}$ is honest. This means that we can use a Byzantine broadcast protocol to have any party $\hat{P}_i \in \mathcal{C}_{\text{BA}}$ consistently broadcast any messages to all the parties in $\mathcal{C}_{\text{BA}}$, i.e., in such a way that the following properties hold:

- (consistency) All partiers in $\mathcal{C}_{\text{BA}}$ output the same message (string) y as their output of the protocol with sender $\hat{P}_i$.
- (validity) If $\hat{P}_i$ is honest, then y is the string that $\hat{P}_i$ intended to broadcast (i.e., his input).

Thus validity implies that for every honest $\hat{P}_i \in \mathcal{C}_{\text{BC}}$, if T_i is the set of valid transactions that $\hat{P}_i$ has seen at round ρ, then every (honest) party in $\mathcal{C}_{\text{BA}}$ will output T_i along with a uniformly random string r_i. Furthermore, by consistency, we know that for every (honest or corrupted) $\hat{P}_j$ the broadcast output with sender $\hat{P}_j$ is the same for all parties in $\mathcal{C}_{\text{BA}}$. Hence the union T of all transactions broadcasted by parties in $\mathcal{C}_{\text{BC}}$ will be the same for all $\mathcal{C}_{\text{BA}}$ members, and the same holds for the concatenation of all random nonces r broadcasted in the current round. Additionally, because the history of the blockchain is the same for every party (in the first round this is the genesis block and in every subsequent round it is the sequence of the blocks until round $\rho - 1$), T_H is the same for every party in $\mathcal{C}_{\text{BA}}$; hence every $\hat{P}_i \in \mathcal{C}_{\text{BA}}$ will compute the same Y in Step 4 of the protocol. Consequently, in Step 5, all honest parties in $\mathcal{C}_{\text{BA}}$ will sign the same (Y, h, ρ) and send it to the parties in $\mathcal{C}_{\text{BC}}$. Since the majority of the parties in $\mathcal{C}_{\text{BA}}$ is honest, this implies that in Step 6, every party in $\mathcal{C}_{\text{BC}}$ will receive (Y, h, ρ) signed by at least the $|\mathcal{C}_{\text{BA}}|/2$ honest parties. Hence, if there is any honest party in $\mathcal{C}_{\text{BC}}$ the transaction pool T of that party broadcasted and included in $\hat{T}$ and will be certifies by by at least $\lceil |\mathcal{C}_{\text{BA}}|/2 \rceil$ signatures from parties in $\mathcal{C}_{\text{BA}}$; this T will be adopted by all parties as the next block, since, as we show below, the adversary is unable to make any party accept any other block. Indeed, with overwhelming probability (i.e., unless the adversary forges an honest party's signature) for any $(Y', h', \rho') \neq (Y, h, \rho)$ the adversary will be able to produce at most $\lceil |\mathcal{C}_{\text{BA}}|/2 \rceil - 1$ signatures on (Y', h', ρ') from parties in $\mathcal{C}_{\text{BA}}$. Hence the no value other than (Y, h, ρ) might be accepted by any party. □

B.4 Proof of Theorem 4

sketch. First we observe that if the reputation system is accurate, then it reflects the probabilities that parties are not corrupted. Hence, the block announcing protocol will always select an honest-majority committee and the block supported by that committee will be added on the PoR-blockchain. Additionally, since the majority in each such committee is honest, the adversary is able to produce valid signatures from less than a 1/2-fraction of the parties in that committee. Hence no party might see any valid conflict on the PoS chain. This proves safety. Block liveness is straightforward since in each slot the honest majority will endorse a new (potentially empty) block. Transaction liveness holds because once a transaction is heard by all honest reputation parties, it will be included in the next slot in which one of the parties in $\mathcal{C}_{\text{BC}}$ is honest. Since the parties in the $\mathcal{C}_{\text{BC}}$ are chosen randomly from an honest majority set $\mathcal{C}_{\text{BA}}$, the probability that in ℓ consecutive slots' $\mathcal{C}_{\text{BC}}$ there is no honest party will be $2^{-\ell|\mathcal{C}_{\text{BC}}|}$.

If on the other hand the reputation system is inaccurate, then we show that no fork can be sustained for more than 2k blocks of the backup PoS-blockchain, as long as the majority of stake is in honest hands. Indeed, assume that the PoR-chain forks at some point (i.e., safety is violated). This means that there are two honest reputation parties $\hat{P}_i$ and $\hat{P}_j$ who see PoR chains that fork on, say, the q-th block of the PoR-chain. The honest majority of stake ensures [DGKR18, BGK$^+$18] that within k blocks, from the round when the q-th PoR-block was created, there will be a block contributed to the PoS-chain by an honest party, say P. This party's block will have to support the view of at most one of $\hat{P}_i$ or $\hat{P}_j$ (whichever is on P's view of the PoR-blockchain). Once this block settles, i.e., enters the common prefix [GKL15, BGK$^+$18]—this happens within in at most $2k$ blocks from round q [BGK$^+$18]—it will be seen by both $\hat{P}_i$ and $\hat{P}_j$ and will trigger a dispute. At this point, the party in $\{\hat{P}_i, \hat{P}_j\}$ who sees his view being disputed will broadcast the block in support of his view. By the transaction liveness of the PoS-chain, this block will be added to the state of the PoS chain within k blocks and will be seen by everyone within another $2k$ blocks. Hence, within $4k$ blocks the fork will be publicly detected. □

C Hoeffdings Inequality

Lemma 5. (Hoeffding's Inequality [Hoe63]) *Let $S = \{x_1, \ldots, x_N\}$ be a finite set of real numbers with $a = \min_i x_i$ and $b = \max_i x_i$. Let $X_1, \ldots, X_n$ be a random sample drawn from S without replacement. Let $\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}$ and $\mu = \frac{\sum_{i=1}^{N} x_i}{N} = E[X_j]$. Then for all $\delta > 0$, $\Pr[\overline{X} - \mu \geq \delta] \leq e^{-\frac{2n\delta^2}{(b-a)^2}}$.*

D Extensions and Future Research

D.1 Countering DDoS Attacks and Minimizing the load of the PoS-chain

Our hybrid protocol $\Pi^{BC}_{\text{PoR/PoS}}$ allows malicious PoS-slot leaders to post fake complains. These complaint will be debunked by the honest parties, but doing so requires costly verification of multiple signatures. In order to avoid such overloading attacks which can lead to DDoS against the backup blockchain and incentivize parties to use it for checking, we will apply the following mechanism inspired by [KZZ16]:

1. The reputation parties will be posting their hash-pointers on the backup PoS-blockchain as part of a collateral-transaction, which commits funds that will be refunded if there is no dispute in the next few rounds (the exact number of rounds and size of collateral will be

specified in the implementation) or there is a dispute which leads to detection of fault in the PoR-blockchain.

2. Similarly, any accuser will have to post, along with his accusation, an analogous collateral-transaction which will be refunded if the dispute leads to detection of fault in the PoR-blockchain.

3. If any dispute is resolved in favor of one of the parties, then this party can claim the collateral of the other and get his own collateral refunded.

The above mechanism will ensure that: (1) reputation parties post the actual hash pointers that they see on the PoR-blockchain—as invalid hashes will lead to disputes resolved in favor of the accuser and, therefore, loss of funds for the accused; and (2) accusers do not post invalid/redundant accusations—as they will, otherwise, lose their collateral.

D.2 Accounting for Dynamic Reputation and Limited Correlations

Our above treatment discusses a limited class of reputation systems, namely correlation-free reputation systems. Recall that this means that the reputation of a reputation-party $\hat{P}_i$ (i.e., $\hat{P}_i$'s probability of becoming corrupted) does not depend on the reputation of other parties $\hat{\mathcal{P}}_j$. The reason for such an assumption is that if one allows for arbitrary correlations in these probabilities, then it is easy to find reputation systems in which every party has probability of not becoming corrupted strictly less than 1/2, yet for any (randomized) sampler the selected corrupted parties will be in the majority with overwhelming probability. An example of such a situation was given in [GIOZ17].

Despite the above strong impossibility result, our mechanism can be applied, either directly or with modifications, to deal with several classes of dependent reputations. This differentiates our definition and use of reputation systems from approaches like [BFK17], which are applied to special distributions. In the following, we discuss some of the distribution classes we can tolerate; a more complete investigation is left as an open research direction.

n-wise independent reputation systems. A wide class of correlated reputation systems which can still be used for our goals is one in which the random variables corresponding to the honesty indicator bits are of n-wise independent for a logarithmic (in the size of the reputation set) n. In fact, our original algorithm will work for such a distribution. This idea of n-wise independence can also be extended to capture additional distributions along the lines of [ALZ13, Theorem 4.3].

Reputation-systems which are approximately correlation-free. The following definition says that a reputation system is δ-correlation-free if and only if it is a δ-close to a correlation-free reputation system.

Definition 4 (δ-correlation-free reputation system)**.** *Let* $\texttt{Rep}$ *be a (potentially non-correlation-free) reputation system for a reputation set* $\hat{\mathcal{P}}$*. For each* $\hat{P}_i \in \hat{\mathcal{P}}$*, let* γ_i *denote the probability, according to* $\texttt{Rep}$*, that* $\hat{P}_i$ *gets corrupted. Let* $\texttt{Rep}_{IND}$ *denote the correlation-free reputation system derived from* $\hat{\mathcal{P}}$*, where each party gets corrupted with probability* γ_i independent of other corruptions*. Then we say that* $\texttt{Rep}$ *is* δ*-correlation-free reputation system if and only if the statistical distance of* $\texttt{Rep}$ *and* $\texttt{Rep}_{IND}$ *is at most* δ*.*

It is straightforward to verify that is δ is very small (negligible in $|\hat{\mathcal{P}}|$) and $\texttt{Rep}_{IND}$ is feasible, then our sampling algorithm discussed above will directly select sets with honest majority. Furthermore, assuming sufficiently many parties with high reputation in $\texttt{Rep}_{IND}$, we can modify our sampling to retain the selection of honest majority even when δ is a small constant. The idea, which will be detailed in the research paper accompanying this white paper, is to move the boundaries of the high-reputation buckets so that any errors that occur by δ-bounded correlations are leveraged by the higher probability of honest parties.

We note in passing, that the above allows us to also capture situations in which the reputation system is inaccurate but only by a bounded amount (e.g., the corruption probabilities that it predicts are off by up to a δ factor from the actual probabilities that the parties get corrupted.) Thus in addition to our fallback security property—which will ensure that as long as the majority of the stake in the system is in honest hands, the blockchain will not fork even with inaccurate reputations—if the number of parties with high (estimated) reputation permits it, then we can design our blockchain so that it is resilient to small inaccuracies in the reputation.

Dynamic (Epoch-resettable) Correlation Systems. Since with an epoch resettable adversary the corrupted set is "reset" at the beginning of each epoch, one might be worried that the adversary across two epochs might be able to use keys he learned in the first of the two epoch in the second one thereby ohlding the keys of a majority of the parties in the latter epoch, even when both epoch's reputation systems are ϵ-feasible.

To counter this scenarios, we consider the following strengthening of the notion of ϵ-feasibility:

Definition 5. *For a reputation system* $\texttt{Rep}$ *for parties from a reputation set* $\hat{\mathcal{P}}$*, a (possibly probabilistic) algorithm* $\texttt{A}$ *for sampling a subset of parties from* $\hat{\mathcal{P}}$*, and an* $\texttt{Rep}$*-adversary* $\mathcal{A}$*, we say that* $\texttt{Rep}$ is strongly $(\epsilon, \texttt{A})$-feasible for $\mathcal{A}$ *if, with overwhelming probability,*[15] $\texttt{A}$ *outputs a set of parties such that at most a* $1/4 - \epsilon$ *fraction of these parties is corrupted by* $\mathcal{A}$*.*

Note that in the above definition, the corrupted parties are chosen according to $\texttt{Rep}$ from the entire reputation-party set $\hat{\mathcal{P}}$, and independently of the coins of $\texttt{A}$. (Indeed, otherwise it would be trivial to always corrupt a majority.)

Definition 6. *We say that a reputation system is strongly* ϵ*-feasible for* $\texttt{Rep}$*-adversary* $\mathcal{A}$*, if there exists a probabilistic polynomial-time (PPT) sampling algorithm* $\texttt{A}$ *such that* $\texttt{Rep}$ *is strongly* $(\epsilon, \texttt{A})$*-feasible for* $\mathcal{A}$*.*

It is easy to verify that if the reputation systems corresponding to all epochs are strongly ϵ-feasible, then the probability that the adversary in two consecutive epochs can launch the above attach (i.e., using the keys learned in the first of the two epochs exceed a minority of corruptions in the second one) is negligible. This is true as long as the two epochs have the same (distribution on the) size of committees, since even if all corrupted parties from the first epoch are selected in the second one, still the adversary will not be able to reach more than a minority of corruptions (since in each epoch at most $1/4 - \epsilon$ fraction is corrupted).

Since in each epoch the adversary needs to release the parties from the previous epoch which are not newly corrupted, we can have the parties refresh their signing keys in each epoch—either by erasing the old keys and posting new ones on the blockchain or by using a forward-secure signature scheme—thereby ensuring that the adversary can never corrupt a majority. It is easy to verify that the above modification ensures security of our construction against epoch-resettable adversaries.

[15] The probability is taken over the coins associated the the distribution of the reputation system, and the coins of $\mathcal{A}$ and $\texttt{A}$.

Capturing additional correlations via a stronger adversary. The epoch-resettable adversary allows for capturing dynamic reputation systems. Nonetheless, when specified by a correlation-free (or δ-correlation-free) system it does not capture situations where reputation parties might be related, e.g, because they are both steered by the same higher level entity.[16] This could be captured by correlated reputation-systems but as discussed above, we cannot tolerate arbitrary correlations. In this section, we discuss how to introduce correlations sufficient for capturing situations like the above, which can still be tolerated.

This is done by what we call a *correlating semi-adaptive* adversary described as follows:

1. Rather than choosing the set of corrupted parties by sampling according to `Rep` at the beginning of each epoch, we allow the adversary to sample multiple times (bounded by his running time) and keep the outcome of his choosing. This means that an adversary that wants a small subset of parties of his choice to be corrupted in Epoch i can achieve this as long as each of these parties has sufficiently large corruption probability.

2. To allow the adversary to further increase his chances of correlating the corrupted sets towards combinations of his choice, we will allow him in this repeated sampling to modify part of the corrupted set by giving up on some of the parties and resampling from the remainder, but with reputations divided by an appropriate factor.

In order, however, to ensure that this adversary cannot target the specific sets of parties that are selected for some round, we require that his sampling from the reputation set occurs by the end of Epoch $i-2$ before the adversary knows the random coins that will be used for sampling the slot-leaders of Epoch i. We remark that this is precisely where the random nonces that were introduced in our Round Broadcast Procedure (Figure 2) are needed.

D.3 Imperfect Randomness

For simplicity in the presentation, we provided a description of our blockchain assuming a randomness beacon for extracting the randomness used in the lottery that defines the slot-committee of each slot. This is a strong assumption but fortunately is it also unnecessary given the standard cryptographic abstraction of hash functions as a random oracle. More concretely, the standard cryptographic assumption under which blockchains are cryptographically analyzed is that the assumed hash function, e.g., SHA256, behaves as a random oracle: On input any new string $s \in \{0,1\}^*$ it outputs a uniformly distributed and unpredictable k-bit string (but if the input has been queried in the past it answers consistently with the previous answer). This is often referred as the *Random Oracle (RO) model*.

Assuming that the genesis block, denoted by $\mathsf{B}_{\mathrm{Gen}}$, contains sufficient amount of entropy, i.e., unpredictable randomness—which we will ensure by including multiple sources of unpredictable yet verifiable randomness, such as the day's weather report or an unlikely event on the newspaper, stock-market values, or blocks from alternative blockchains—we can use such a hash function in the RO model to extract the randomness we need for each epoch as follows:

- For static reputation system and adversaries, we can define the randomness of the i-th epoch to be $\mathsf{h}(\mathsf{B}_{\mathrm{Gen}}||i1)||\mathsf{h}(\mathsf{B}_{\mathrm{Gen}}||i2), \ldots$ (where $||$ denotes concatenation of strings). Note that although this means that the adversary will be able to predict all slot committees by just knowing the genesis block, a static reputation-limited adversary is forced to sample his corrupted parties

[16] For example, if the reputation parties are public figures, such as popular artists, they might follow their agent's directions.

set only once (at the beginning of the protocol) and according to the static reputation system which, by the unpredictability of B_{Gen}, will be sufficiently independent of the output of h. We note in passing that in this case we do not even need the randomness which is explicitly introduced in each epoch to prove security of the protocol. However, the static reputation and adversary assumption yields a too weak adversary. Therefore, in the following we discuss how to cope with stronger adversaries.

- The above simple randomness-extraction idea still works for dynamic reputation systems and epoch-resettable adversaries, as long as the reputation system ensemble is a sequence of independent (static) reputation systems. This however excludes situations where malicious parties might try to manipulate their reputation (e.g., by temporarily playing honestly) depending on the protocol's history. To model also such situation we modify the way this randomness is selected as follows: As discussed in Figure 2, each block B_i in the blockchain includes a hash pointer $\mathsf{h}(\mathsf{B}_{i-1})$ to the previous block.[17] The randomness for each epoch i is then decided by the value of this hash pointer at a slot of the previous epoch which is sufficiently deep in the chain to ensure that the corresponding block has reached every (honest) party[18] but not too deep, so that an adversarial reputation party can considerably correlate his reputation with it.

  We note that as the adversary might be choosing the randomness (nonces) he contributes to each block potentially depending on honest slot-committee members' nonces, he still has the power to affect the outcome of the hash function (RO). However, because, by design of the protocol in Figure 2, the concatenation of all nonces—including honestly generated—contributed in each slot will be used to form the slot randomness r_ρ, we can argue, analogously to [KRDO17, BGK$^+$18], that this effectively corresponds to allowing the adversary to resample from the distribution of the RO's output up to polynomially many times, and choose his favorite outcomes of this iterated-sampling procedure as the value adopted for the current epoch randomness—this process is often referred to as *randomness grinding*. Similarly to [KRDO17, BGK$^+$18], we can argue that as long as the range of $\mathsf{h}(\cdot)$ is exponentially big—which is true for all common hash functions—such grinding might only increase the success probability of the adversary by a negligible amount. In fact, the above idea can be extended to also tolerate correlating semi-adaptive adversaries, as above. The idea is that this extra power of the adversary effectively increases his randomness grinding ability by a polynomial factor, which, however, is not sufficient to considerably affect the adversary's overall success probability (it will remain negligible).

D.4 Relaxing Synchrony

We next discuss how to revise protocol $\Pi^{BC}_{\text{PoR/PoS}}$ and its analysis to remove the arguably strong simplifying assumptions of perfect synchrony with zero-delay channels. We discuss how to remove each assumption separately in the following two subsections.

Partially Synchronous Network We start by replacing the zero-delay multicast network, aka instant delivery multicast, with Δ-*delay multicast* network which captures what happens in real world diffusion networks. In such a network, when a message is inserted by some honest party,

[17] As usually in the blockchain setting the transaction will be identified by means of a Merkle-Tree; and for efficiency, rather than the whole block B_i, the hash pointer will include the root of the tree along with the header (metadata) of the block—the metadata explicitly includes the current nonce r.

[18] This becomes relevant when the network is not perfect, but it has delays (cf. Section D.4).

it is guaranteed to be delivered to every honest party within Δ rounds. We note that Δ is an upper bound on the delay but the adversary is able to deliver submitted messages in any order and/or incur any delay up to Δ rounds from the time the message was inserted into the network. As shown in [BMTZ17], in the synchronous model this network can be constructed from simpler unicast channels of the type used in all mainstream blockchains.

Our blockchain approach can be adapted to this setting using the technique from [CCGZ16, CCGZ17]. This technique allows to design primitives assuming zero-delay channels, and prove their security under this assumption, and then automatically compile the protocol to work for the setting of bounded-delay channels, in a way that preserves both the number of rounds and the security.

Removing the Global Clock Our design and analysis assumes a global clock. This is a realistic assumption as NTP offers such a functionality to Internet protocols. As part of future research, we will combine our approach with techniques in clock synchronization [DHS84, LMS84, HSSD84, DW95, DW93, AHR93, ST85, LLW08, OPS99] to ensure that our protocol can operate without the global clock assumption, but under the weaker assumptions that parties have local clocks that advance at roughly the same speed. In addition to the above, we will take advantage of the back-up proof-of-stake blockchain to establish clock synchronization under the honest-majority of stake assumption, using [BGK$^+$19], and/or combine it with the traditional synchronizers discussed above to get a best-of-both-worlds synchronizer.

Reducing the Number of Rounds Our round broadcast protocol in Figure 2 uses an off-the-self broadcast protocol among the slot committee. Instantiating this with a traditional deterministic broadcast protocol will incur a number of rounds which is, in the worst case, linear to the slot-committee size. To make sure that, on average, the actual execution time is very short, requiring a small constant number of rounds, we will make use of the expected constant round protocols from [FM88, FM85, KK06, GKKO07, Mic17] instead. We note in passing that in addition to these protocols being very fast on expectation, they are almost certain to terminate within a sufficiently short (polylogarithmic in the slot-committee size) number of rounds. Thus, the resulting protocol will be very fast on average and sufficiently fast in the worst case.[19] We also remark that the composition (sequential and or parallel execution) of such protocols with themselves and with non-zero-delay networks as the one used here is known to have several pitfalls. Thus will can again use the compiler from [CCGZ16, CCGZ17] to make sure that the resulting protocol will retain its security.

D.5 Establishing, Updating, and Extending the Reputation System

The current design assumes an already established reputation system and ensures that as long as the reputation system is sufficiently accurate—i.e., the reputation of a reputation party is close to his probability of being honest—the protocol will behave according to its abstract specification, i.e., it will securely realize a decentralized transaction ledger. Thus our cryptographic analysis is orthogonal to the mechanism that establishes, updates, and extends the reputation system. Nonetheless in this section we discuss a number of directions and associated research questions that relate to the deployment and rational analysis of the system.

The incentive structure. By design, $\Pi^{BC}_{\text{PoR/PoS}}$ assumes a very simple incentive structure to characterize the reputation parties' rationality: Reputation parties should always prefer higher

[19] The probability that these protocols terminate after round ρ is roughly $2^{-\rho}$.

reputation values. There are a number of approaches one can take to enforce and analyze such incentives. Although the complete game-theoretic design and analysis is beyond the scope of this current paper, it is a very interesting future research problem; in the following we discuss some choices and their intuitive consequences.

A mechanism to incentivize reputation parties to increase their reputation and behave according to it is to create a cryptocurrency on top of our blockchain and tie the parties' reputation with their actual stake in the system. In our original design, we will require reputation parties to lock part of their stake as collateral; we will also enforce a very minor fee to the parties whose proposal is included in the blockchain that they will be able to either add to their collateral, thereby increasing their reputation—hence also their influence on the system and their probability of getting more fees—or they will be able to receive it as an actual reward in coins.

The value of the fee will be carefully computed to be diminishing with the reputation of the party in a way that marginally increases the parties' expected gain—e.g., parties with higher reputation achieve lower fees every time they win, i.e., get selected for a slot committee, but still higher total fees on average (recall that higher reputation parties win more often). We remark that, unlike generic proof-of-stake blockchains, typical cases of reputation parties will be publicly recognizable entities, e.g., famous artists, banks, universities, companies, etc., so that one cannot manipulate the above mechanism by creating multiple identities. Additionally, the following mechanism can ensure that reputations are not artificially inflated[20]: Periodically, an amortization factor will be applied to all reputations, so that their relative value changes marginally, but their absolute value is preserved within limits that represent probabilities, i.e., stays between 0 and 1.

In addition to the above mechanism, our reputation-based blockchain as discussed in the introduction, has the potential to disrupt the recommendation systems industry which currently uses opaque algorithms to decide recommendations and rankings. An interesting research direction in the intersection of AI and cryptography/security is to take as input to the reputation calculation the rankings of existing recommendation systems, e.g., Amazon, Yelp, Ebay, etc., to develop a universal, transparent, and decentralized recommendation system running on our blockchain. The high throughput of our protocol will be instrumental in linking on-chain pointers to potentially externally stored, yet immutable reviews.

Modifying the reputation according to the parties' behavior. As discussed above, a mechanism taking advantage of transaction fees and rewards will be associated with the parties' reputation to ensure that parties prefer to increase their reputations. Additionally, the following penalization mechanism will be applied to ensure that reputation parties with the above preferences: (1) are sufficiently available, and (2) do not attempt to disrupt the systems consistency (and create forks):

- Reputation parties will declare availability for future slots and their rewards and/or reputation will increase the more available they make themselves (i.e., availability will be taken in consideration in calculating the associated rewards and reputation) but will decrease more radically if they declare themselves available but do not participate. We remark that this second property is trivial to ensure assuming instant-delivery channels—as everyone can decide in round ρ whether or not a party sent its round $\rho - 1$ message—but it is tricky in the bounded-delay network, as a party might have sent its message, which is delayed (by the adversary) in the network. One can rectify this by waiting sufficiently long—at least as much

[20] If everyone plays honestly then all reputations increase and will eventually become larger than 1, which is a degenerate situation we want to avoid, as reputations correspond to probabilities.

as the network delay Δ—to decide on the reputation, but this incurs the risk of a malicious party abstaining in the protocol and then quickly delivering a message for the reputation update mechanism. Instead we will use the blockchain itself to implement a voting protocol for parties to ensure that a message was contributed by a reputation party on the slot it volunteered for. Although not directly related with the cryptographic design of the system, the game-theoretic analysis of such a mechanism is part of our team's research agenda.

- If any party presents on the blockchain verifiable evidence of some reputation-party cheating, e.g., two conflicting signatures from that party for the same slot, then the reputation of that party is set to 0 by anyone that views this evidence. This means that the related reputation party is effectively removed from the reputation set.

DDoS Effect and Mitigation Our blockchain leverages the reputation system to ensure that it selects an honest-majority committee in every slot. However, as discussed here, the identities of slot leaders become public prior to the round in which they participate in the committee. One might argue that this opens the system to DDoS attacks, as even honest parties might be targeted. We point out however that that any such attack will result in an empty slot and not a fork, as the adversary, even when DDoS-ing a party cannot create a majority of signatures from the current slot's committee. In fact, one can mitigate this issue by either using—e.g., reputation-penalties for successful DDoS attacks—which will inceltivize high reputation parties to safeguard their availability, or using verifiable random functions (VRF) as in [DGKR18, BGK$^+$18, GHM$^+$17] but this will require not only circulating more information (VRF proofs) but also flooding it to the entire network. Alternative one can spread out VRF slots will help recover from stalling due to very effective DDoS attacks, but still most slots will be using the reputation-based incentives, thereby ensuring the good communication efficiency of our blockchain protocol. The exact dynamics of this mechanism are left as a future research direction.