Proof-of-Reputation Blockchain with Nakamoto Fallback

Leonard Kleinrock[1], Rafail Ostrovsky[1], and Vassilis Zikas[2]

[1] Computer Science Department, UCLA, {lk,rafail}@cs.ucla.edu
[2] School of Informatics, University of Edinburgh, vzikas@inf.ed.ac.uk

Abstract. Reputation is a major component of trustworthy systems. However, the subjective nature of reputation, makes it tricky to base a system's security on it. In this work, we describe how to leverage reputation to establish a highly scalable and efficient blockchain. Our treatment puts emphasis on reputation fairness as a key feature of reputation-based protocols. 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. To avoid potential safety and/or liveness concerns stemming from the subjective and volatile nature of reputation, we propose a hybrid design that uses a Nakamoto-style ledger as a fallback. To our knowledge, our proposal is the first cryptographically secure design of a proof-of-reputation-based (in short PoR-based) 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 fallback blockchain does not fail.

Keywords: Blockchain, proof of reputation, Byzantine agreement

1 Introduction

Many decisions taken in modern society are based on reputation: Fans are likely to follow suggestions from their idols, social network followers are likely to adopt suggestions of the friends and groups, and people often use publicly available ranking systems such as E-Bay, Yelp, AirBnB, Amazon, etc to make decisions regarding online-shopping, choice of vacation, accommodation, eating out, insurances, investment, medical, etc. In this work we leverage the power of reputation to establish a reliable, permissionless blockchain. Our design assumes that certain parties in the system have a reputation-rank, which is recorded on the blockchain itself. Similar to public ranking systems as above, where the "stars" of a party are interpreted as a prediction of the quality of its offered services, the interpretation of our reputation-ranks is that the higher a party's reputation the higher are the chances that the party will behave honestly in maintaining the blockchain.

In a nutshell, our design goals are two-fold: (1) To rely on the reputation system to build a simple, scalable decentralized ledger with optimized finality, communication, and computation, and with a formal proof that relates the security of the protocol to the quality of the reputation system; importantly, our PoR-blockchain aims to satisfy a new intuitive notion of participation fairness that promotes inclusivity. (2) To address the subjectivity of reputation as a resource, by backing our blockchain's safety and liveness with a fallback mechanism that ensures that even if the reputation estimate is severely flawed, our protocol does not create long forks.

Our Results. We devise a hybrid blockchain-ledger design which is primarily based on reputation but uses a Nakamoto ledger as fallback. We use the term *Nakamoto ledger* similar to [4] to refer to blockchain-based ledger protocols that follow the eventual consensus paradigm (e.g., Bitcoin, Ouroboros, etc) and realize a ledger as described in [5,3]. For the purpose of exposition, in this work, we focus on the fallback ledger being proof-of-stake-based and tune our analysis accordingly—we refer to this paradigm as *PoR/PoS-hybrid.* design However, our treatment can be extended to other types of fallback blockchains (even those that do not follow the Nakamoto paradigm) under their respective assumptions. In the following we provide an overview of our design and discuss its properties in comparison to existing approaches.

PoR-blockchain We assume 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 securely 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*).[3]

Informally, a reputation system for a party set $\hat{\mathcal{P}}$ is similar to a *probabilistic adversary structure* [20]: It assigns to each subset of $\hat{\mathcal{P}}$ a probability that this subset is corrupted by the adversary—we 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 $|\hat{\mathcal{P}}|$ independent boolean random variables. The probability that the ith variable is 1 is the probability that the i-th party in $\hat{\mathcal{P}}$—in a given canonical ordering, e.g., using the party-IDs—is honest. This is similar to *independent reputation systems* investigated in [1]. We also extend this 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 2) 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 is analogous to the feasibility condition from [1].)

Our ledger protocol proceeds in rounds. Each block is associated with a *slot*, where a slot lasts a predefined number of rounds. We use the reputation system as follows (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 common randomness used in the first epoch. 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. (For dynamic reputation, the contents of an epoch are used to extracts coins for the following epoch.)

The above implicit lottery is used to elect a *slot committee* $\mathcal{C}^i_{\mathrm{BA}}$ for each slot i, which is responsible for gathering all known transactions at the beginning of the slot that have not yet been inserted in the blockchain, and proposing the block corresponding to this slot along with evidence to enable all parties to agree on this block. More concretely, in every slot, every party in a random subset $\mathcal{C}^i_{\mathrm{BC}}$ of $\mathcal{C}^i_{\mathrm{BA}}$ pools all new and valid transactions received by the blockchain users into a set, and broadcasts this set to $\mathcal{C}^i_{\mathrm{BA}}$, by means of a byzantine broadcast protocol. The union of the tractions broadcasted by $\mathcal{C}^i_{\mathrm{BC}}$ is then signed by every member of $\mathcal{C}^i_{\mathrm{BA}}$ and circulated to the whole 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 $\mathcal{C}^i_{\mathrm{BA}}$. As long as for each of the slot committees the majority of its members is honest—a property which will be induced by an accurate and feasible reputation system—the above consensus protocol will achieve agreement on the proposed block.

Of prime importance in our construction, and a novelty of our PoR-ledger, is a mechanism which ensures an intuitive notion of inclusivity. For clarity, we focus our description and analysis on static correlation-free reputation systems. Different ways to extend our result to dynamic and correlated adversaries are discussed in the appendix. As proved in [1], for any feasible static correlation-free reputation system, the following sampler, denoted as $\mathtt{A}_{\max}$, outputs a committee with honest majority: Order the parties according to their reputation and choose a sufficient number (cf. [1]) of reputation parties with the highest reputations. The above simple $\mathtt{A}_{\max}$ algorithm is optimal (up to negligible error) for minimizing the risk of electing a dishonest-majority committee, but it suffers from the following issues: First, if some malicious party establishes (e.g., by behaving honestly) a high reputation, then this party will be included in almost every committee. Second, the approach lacks a natural *fairness* property which would give *all* parties voting power according to their reputation (even to parties with low reputation). Such a fairness property is important for sustainable decentralization, as it does not deter participation of parties with low reputation making the overall system more inclusive. [4] We propose and instantiate an appropriate notion of reputation fairness, termed *PoR-fairness*, which addresses the above concerns. The idea behind PoR-fairness is that every reputation party should get a "fair" chance to be part of each slot-i committee $\mathcal{C}^i_{\mathrm{BA}}$. Defining such a notion turns out to be non-trivial (cf. Section 3.1). Intuitively, a reputation-based lottery (i.e., committee selection protocol) is

[3] 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, Ethereum, Ouroboros, Algorand, etc.

[4] For instance, one can consider a mechanism which rewards honest behavior by increasing the parties' reputation.

reputation-fair, if, (1) on average, the representation of parties on the selected committee becomes higher, the more likely those parties are to be honest (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 also turns out to be a challenging task and a core technical contribution of our work.

The Nakamoto Fallback Arguably, reputation is a subjective and manipulable resource. For instance, the way reputation of new parties is assigned might be flawed or a malicious party might act honestly to build up its reputation, and then use its earned trust to attack the system. We address this in the following way: We back our PoR-blockchain with a light-weight use of a Nakamoto-style blockchain[5] which will ensure that if the reputation system fails to provide security, this will be observed and agreed upon on the Nakamoto-chain. This ensures that reputation parties, who try to actively cheat (e.g., sign conflicting messages) or abstain to hurt liveness or safety, will be exposed on the fallback chain.

The idea for our fallback is as follows: Parties report on the fallback chain (an appropriate hash of) their view of the PoR-blockchain. If any party observes an inconsistency of this with its own view, it contest the reported view, by posting an accusation along with appropriate evidence (signatures and hash-pointers). The original reporting party is then expected to respond to this accusation by the contents and signatures of the disputed block. Once the accusation is answered, it will either expose one of the two disputing parties as a cheater, or it will expose a misbehavior on the PoR-chain (e.g., signing conflicting messages). In either case the exposed party gets its reputation zeroed out and is excluded from the system.

This fallback mechanism fortifies the security of the blockchain under an independent backup assumption, e.g., majority of honest stake. Note that the fallback blockchain is not used for communicating transactions, but only digests (hashes) of blocks from the main, reputation chain as discussed below. Furthermore, it does not need to run in a synchronized manner with the main PoR-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 design as a *PoR/PoS-Hybrid Blockchain*. We remark that the generic fallback mechanism allows to recover from safety attacks. By an additional assumption on the quality of the reputation system we can also prevent liveness attacks as discussed in Section 5.

Properties of our construction. The proposed reputation-based blockchain takes advantage of the nature of the reputation-system to improve on several properties of existing constructions as discussed below. Provided that the reputation is accurate, the parties will enjoy such improvements. As is the case with any assumption on resources, it is impossible to know a priori if the underlying assumption, in our case accuracy of the reputation, is true. However, unlike constructions which completely fail when their core assumption is false (e.g., dishonest majority of stake in PoS) our fallback mechanism will ensure that even if our primary assumption, i.e., accuracy of the reputation, is violated, still the basic security properties are not (or any violation is swiftly detected) as long as the secondary, fallback assumption holds. This yields a natural optimistic mechanism for use of resources as discussed below. In the following we discuss the advantages that our protocol in this optimistic mode, i.e., under the primary assumption that the reputation system is accurate.

Efficiency and Scalability: A reputation system induces probabilities that a parties behaves maliciously, which, technically, provides information to the protocol designer (and parties) on the corruption capabilities of the adversary. This allows us to avoid the need for communication- and setup-heavy adaptive security tools, e.g., compute and communicate proofs of verifiable random functions.[6] Additionally, it allows us to associate reputation parties with public keys and public physical IDs, e.g., public IP address, which means they can communicate through direct point-to-point channels rather than diffusion/gossip network. This yields both concrete and asymptotic improvements. First, depending on the network topology, this can improve the overall concrete message complexity and yield denial-of-service (DoS) attack protection in practice—open

[5] As discussed above, here we focus on a proof-of-stake Nakamoto-style blockchain, e.g., [28], but our fallback uses the Nakamoto blockchain in a blackbox manner and can therefore be instantiated using any blockchain that realizes a Bitcoin-style transaction ledger [5].

[6] As a side note, our blockchain does address concerns about adaptivity in corruptions through its fallback mechanism, which can be adaptively secure.

diffusion networks are more susceptible to DoS. Additionally, as most communication occurs only among the (polylogarithmic) size slot committees and between these committees and the player set, even ignoring the overhead of gossiping, the overall message complexity per slot is $O(n \log^{\epsilon} n)$ as opposed to the $\Omega(n^2)$ complexity of standard blockchains relying solely on flooding.

High Throughput (Transaction-Liveness): Existing solutions implicitly assign to each block one effective-block proposer [28,14,3,22]—multiple parties might win the lottery but only the proposal of one is adopted. Instead, in our PoR-blockchain a (small) committee $\mathcal{C}_{\text{BC}}$ of proposers is chosen in each slot, and the union of their transactions-views is included. To ensure that honest transactions are included in a block it suffices that *any* of the block proposers in the corresponding $\mathcal{C}_{\text{BC}}$ is honest. This will be true with probability at least $1 - 1/2^L$, where L is the size of $\mathcal{C}_{\text{BC}}$, as we are choosing L parties out of $\mathcal{C}_{\text{BA}}$ which has honest majority. In comparison, in systems that choose one proposer per block, this probability is upper-bounded by (roughly) t/n, where t is the number of corrupted parties/resources (e.g., the amount of adversarially owned stake) and n is the total amount of resources. The above discrepancy can be impactful in situations where the transaction-submitting mechanism has high latency—e.g., due to bad/restricted access to the blockchain network, or some points to the blockchain network are unreliable.

Finality: Since the parties decide on the next block by means of a Byzantine broadcast protocol, agreement is achieved instantly by the end of the slot. This is similar to standard synchronous BFT-based blockchains, and is in contrast to Nakamoto-style blockchains [34,9,28] which achieve eventual consistency, aka the common prefix property [21,37]. We stress that this is the case assuming the reputation system is accurate. One can argue that this might be an insufficient guarantee as to get full confidence and some users might want to wait for the situation to settle also on the fallback chain—i.e., ensure that no honest party contests their view. Nonetheless, it allows for a tiered use of the assumptions, which naturally fits situations where different transactions have different degree of importance, as is for example the case in cryptocurrencies: For small-amount transactions the users can trust the PoR-blockchain and consider the transaction settled as soon as it is finalized there. The more risk-averse users (or users with high-stake transactions) can wait for the backup chain to confirm that there is no accusation. The above mode is the natural blockchain analog of how reputation is used in reality: If a service or an investment is recommended by a highly reputable source, then it typically enjoys higher trust. However, for risky actions, the actors usually seek further assurances that might take longer time.

Related Literature. Asharov et al. [1] defined reputation systems for multi-party computation (MPC) and proved necessary and sufficient conditions on 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 with reputation were previously made in the context of blockchains and cryptocurrencies. None of these attempts 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. Additionally many of these works lack a protocol specification, security model and proofs, and often even a credible security argument [19], and/or rely on complicated reputation mechanisms and exogenous limitations on the adversary's corruption power [11]. 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 [40].

Notably, [8] proposed a reputation-module which can build a scalable blockchain on top of a BFT-style consensus protocol, e.g., PBFT or Honey Badger [33]. The idea is that this reputation module can be used by the parties to select smaller next round committees. 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. Instead our PoR-blockchain construction is secure under the assumptions of accuracy of the reputation system, irrespective of the properties of the fallback blockchain. The result from [8] also proposed a notion of reputation-fairness, which renders a reputation-based lottery more fair the closer its outcome is to a uniform sample. This notion of fairness seems unsuitable for our goals, 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? And how would this incentivize parties to build up their reputation? Our fairness definition addresses this concern, at a very low overhead.

Hybrid blockchains which use an alternative consensus mechanism as a fallback were also previously used in Thunderella [38] and Meshcash [7]. Their protocols rely on smart refinements of the proof-of-work and/or proof-of-space-time paradigms, and uses novel methods 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 [32] devises a finality layer module on top of a proof-of-stake blockchain. Their construction achieves fast 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.

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 $s' \in \{0,1\}^{k-D}$.

Organization of the Remainder of the Paper. After discussing our model in Section 2, in Section 3 we define and instantiate a reputation-fair lottery. Section 4 describes a PoR-based blockchain-ledger protocol for static reputation systems, and Section 5 describes the hybrid PoR/PoS ledger protocol.

2 The Model

Our ledger protocol is among a set of parties $\mathcal{P} = \{P_1, \ldots, P_n\}$. A subset $\hat{\mathcal{P}}$ of the parties, called *reputation parties*, have a distinguished role in the protocol, and are tasked with proposing and confirming blocks to be added in the blockchain. To avoid overcomplicating our description, we describe our protocol here for a static set of parties; in the appendix we discuss how this set can be dynamically updatable, similar to the dynamic participation of [28,3]. As is common in the blockchain literature, our statements are in the random oracle model, where a hash function is assumed to behave as a random function. As a setup, we assume that all parties have the genesis block which includes sufficient initial randomness and the reference reputation-system. In terms of cryptographic assumptions we will assume existentially unforgeable digital signatures [23] 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.

Communication and Synchrony. We assume synchronous communication, where messages sent by honest parties in some round are delivered by the beginning of the following round, and a rushing adversary [10]. The protocol timeline which is divided into *slots*, where each slot consists of a fixed number of rounds. An *epoch* consist of a predefined number of slots. We assume a cryptographic adversary who gets to *actively* corrupt parties—i.e., takes full control over them.

Parties have two means of communicating. (1) A diffusion (multicast) network available to everyone in $\mathcal{P}$, build by means of a standard flooding/gossiping protocol over a (potentially incomplete but) connected communication graph of unicast channels [5]—this is similar to [34,5], Ethereum [9], Cardano/Ouroboros [28,3], Algorand [22], Thunderella [38], etc. For simplicity, we abstract this network by means of a zero-delay *multicast* primitive (cf [5]): 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.[7] We note that one can use techniques from

[7] 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) in the round when this forwarding occurs.

the blockchain literature [3,22,22] to relax this *perfect* synchrony assumption—at the cost of a stricter feasibility condition on the reputation system. A discussion of such a relaxation is included in the appendix. (2) The second type of communication is among reputation parties. These parties have known physical identities (e.g., IP addresses) and can communicate through direct channels, without flooding (e.g., via TCP/IP). We remark that existence of such channels 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 do exist and are used then can considerably reduce the traffic and yield a more scalable/efficient solution, as discussed in the introduction.

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 $(h_1, \ldots, h_m) \in \{0,1\}^m$.[8] 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. 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 [1]. 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. Such dynamic reputation systems are useful in an evolving and reactive primitive such as a ledger protocol, where the reputation of parties might change depending on their behavior in the system and/or other exogenous factors.

We focus on the setting where each h_i corresponds to the output of an *independent* indicator random variable H_i, i.e., 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., $\Pr[H_i = 1] = \mathtt{R}_i$).[9] 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.

The adversary's corruption capabilities are specified by the reputation system. A *static* reputation-bounded adversary for reputation system $\mathtt{Rep}$, also referred to as a static $\mathtt{Rep}$-*adversary*, corrupts the set of parties at the beginning of the protocol according to $\mathtt{Rep}$, and sticks to this choice. In particular, 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 for each $h_i = 0$ the reputation party $\hat{P}_i \in \hat{\mathcal{P}}$ is corrupted by the adversary. For dynamic reputation systems, a stronger type of adversary, which we call *epoch-resettable* adversary corrupts a completely new set of parties at the beginning of each epoch, according to the reputation system at the beginning of that epoch—this is similarly to mobile adversaries [36]. Here we focus our analysis to the static case; an extension to epoch-resettable adversaries is discussed in the appendix.

3 Reputation-based Lotteries

At the heart of our construction is a lottery that chooses a (sublinear) set of parties according to their reputation. To demonstrate the idea, let us 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 $\mathtt{R}_i > 0.5 + \epsilon$ for a constant ϵ, e.g,. $\mathtt{R}_i > 0.51$.

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. In Scenario 2, on the other hand, the situation is different. Here, by choosing a polylogarithmic

[8] 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.

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

random committee we can guarantee (except with negligible probability)[10] that the majority of those parties will be honest (recall that we assume independent reputations), and we can therefore employ a consensus protocol to achieve agreement on each transaction (block).

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,*[11] A *outputs a set of parties such that at most a* $1/2 - \epsilon$ *fraction of these parties is corrupted by* $\mathcal{A}$*.*

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.)

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 the (expected) number of honest parties in the committee it suffices to always choose the parties with the highest reputation. In fact, [1] generalized this to arbitrary correlation-free reputation systems by proving 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. [1]) parties with the highest reputation induces a set which has honest majority. We denote this algorithm by $\mathtt{A}_{\max}$.

Lemma 1 ([1]). *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* $\mathcal{A}$*.*

As discussed in the introduction, despite 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 an algorithm for achieving it.

3.1 PoR-Fairness

As a warm up, let us consider a simple case, where all reputations parties can be partitioned in two subsets: $\hat{\mathcal{P}}_1$ consisting of parties with reputation at least 0.76, and $\hat{\mathcal{P}}_2$ consisting of parties with reputation between 0.51 and 0.75. Let $|\hat{\mathcal{P}}_1| = \alpha_1$ and $|\hat{\mathcal{P}}_2| = \alpha_2$. We want to sample a small (sublinear in $|\hat{\mathcal{P}}| = \alpha_1 + \alpha_2$) number y of parties in total.

Recall that we want to give every reputation party a chance (to be part of the committee) while ensuring that, the higher the reputation, the better his relative chances. A first attempt would be to first sample a set where each party $\hat{P}_i$ is sampled according to his reputation (i.e., with probability $\mathtt{R}_i$) and then reduce the size of the sampled set by randomly picking the desired number of parties. This seemingly natural idea suffers from the fact that if there are many parties with low reputation—this is not the case in our above example where everyone has reputation at least 0.51, but it might be the case in reality—then it will not yield an honest majority committee even when the reputation system is feasible.

A second attempt is the following. Observe that per our specification of the above tiers, the parties in $\hat{\mathcal{P}}_1$ are about twice more likely to be honest than parties in $\hat{\mathcal{P}}_2$. Hence we can try to devise a lottery which selects (on expectation) twice as many parties from $\hat{\mathcal{P}}_1$ as the number of parties selected from $\hat{\mathcal{P}}_2$. This will make the final set sufficiently biased towards high reputations (which can ensure honest majorities) but has the following side-effect: The chances of a party being selected diminish with the number of parties in his reputation tier. This effectively penalizes large sets of high-reputation parties; but formation of such sets should be a desideratum for a blockchain protocol. To avoid this situation we tune our goals to require that when the higher-reputation set $|\hat{\mathcal{P}}_1|$ is much larger than $|\hat{\mathcal{P}}_2|$, then we want to start shifting the selection towards $\hat{\mathcal{P}}_1$. This leads to the following informal fairness goal:

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:

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

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

1. $x_1 + x_2 = y$
2. $x_1 = 2\max\{1, \frac{\alpha_1}{\alpha_2}\}x_2$ *(representation fairness)*
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 $\hat{\mathcal{P}}_i$ *(non-discrimination)*, but parties in $\hat{\mathcal{P}}_1$ are twice as likely to be selected as parties in $\hat{\mathcal{P}}_2$ *(selection fairness)*.

Assuming α_1 and α_2 are sufficiently large, the above goal can be achieved by the following sampler: For appropriately chosen numbers ℓ_1 and $\ell_2 \geq 0$ with $\ell_1 + \ell_2 = y$, sample ℓ_1 parties from $\hat{\mathcal{P}}_1$, and then sample ℓ_2 parties from $\hat{\mathcal{P}}_1 \cup \hat{\mathcal{P}}_2$ (where if you sample a party from $\hat{\mathcal{P}}_1$ twice, replace him with a random, upsampled party from $\hat{\mathcal{P}}_1$). As it will become clear in the following general analysis, by carefully choosing ℓ_1 and ℓ_2 we can ensure that the conditions of the above goal are met. For the interested reader, we analyze the above lottery in Appendix A.1. Although this is a special case of the general lottery which follows, going over that simpler analysis might be helpful to a reader, who wishes to ease into our techniques and design choices.

Our PoR-Fairness Definition and Lottery We next discuss how to turn the above informal fairness goals into a formal definition, and generalize the above lottery mechanism to handle more than two reputation tiers and to allow for arbitrary reputations. To this direction we partition, as in the simple example, the reputations in $m = O(1)$ tiers[12] as follows: For a given small $\delta > 0$, the first tier includes parties with reputation between $\frac{m-1}{m} + \delta$ and 1, the second tier includes parties with reputation between $\frac{m-2}{m} + \delta$ and $\frac{m-1}{m} + \delta$, and so on. Parties with reputation 0 are ignored.[13] We refer to the above partitioning of the reputations as an *m-tier partition.*

The main differences of the generalized reputation-fairness notion from the above informal goal, is that (1) we parameterize the relation between the representation of different ties by a parameter c (in the above informal goal $c = 2$) and (2) we do not only require an appropriate relation on the expectations of the numbers of parties from the different tiers but require that these numbers are concentrated around numbers that satisfy this relation. The formal reputation fairness definition 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 tiers as above (where the parties in $\hat{\mathcal{P}}_1$ have the highest reputation) 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 $\textsc{Member}_j$ 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 $(\hat{P}_{i_1}, \hat{P}_{i_2}) \in \hat{\mathcal{P}}_i \times \hat{\mathcal{P}}_{i+1}$, 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 $\hat{P}_{i_1}, \hat{P}_{i_2}$ 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.

[12] This is analogous to the rankings of common 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.

[13] This also gives us a way to effectively remove a reputation party—e.g., in case it is publicly caught cheating.

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 ℓ_1 parties from the highest reputation set $\hat{\mathcal{P}}_1$, then we sample ℓ_2 parties from the union of second-highest and the highest $\hat{\mathcal{P}}1 \cup \hat{\mathcal{P}}_2$, then we sample ℓ_3 parties from the union of the three highest reputation tiers $\hat{\mathcal{P}}_1 \cup \hat{\mathcal{P}}_2 \cup \hat{\mathcal{P}}_3$, and so on. As we prove, the values ℓ_1, ℓ_2, ℓ_3 etc. can be carefully chosen so that the above fairness goal is reached whenever there are sufficiently many parties in the different tiers. We next detail our generalized sampling 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*[14]—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 h which we will assume behaves as a random oracle.[15] 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 $\log \frac{|\mathcal{P}|}{t}$ 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. The second sampler denoted by $\mathsf{RandSet}$ is the straight-forward way to sample a random subset of t parties from $\mathcal{P}$ 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 $\mathcal{P}$. For completeness we have included detailed description of both samplers in Appendix A.2. Given the above two samplers, we can provide the formal description of our PoR-fair lottery, see Figure 1. Theorem 1 states the achieved security.

Theorem 1 (Reputation-Fair Lottery for $m = O(1)$-tiers). *In the above lottery $\mathsf{L}(\hat{\mathcal{P}}, \mathsf{Rep}, (c_1, \ldots, c_m), \delta, \epsilon, r)$, 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 number of parties in the final committee that are from set $\hat{\mathcal{P}}_i$; and for each $i \in [m]$ 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 Rep is ϵ_f-feasible for a static Rep-bounded adversary $\mathcal{A}$, then for the set $\mathcal{P}_{\mathtt{sel}}$ of parties selected by L the following properties hold with overwhelming probability (in the security parameter k):*

1. $|\mathcal{P}_{\mathtt{sel}}| = \Theta(\log^{1+\epsilon} n)$
2. *for some constant $\epsilon_\delta > 0$ adversary $\mathcal{A}$ corrupts at most an $1/2 - \epsilon_\delta$ fraction of the parties in $\mathcal{P}_{\mathtt{sel}}$*
3. *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 D.1. In the following we included a sketch of the main proof ideas.

Proof (sketch). We consider two cases: Case 1: L noes not reset, and Case 2: L resets.

In Case 1, The lottery is never reset. This case is the bulk of the proof.
First, in Lemma 3 using a combination of Chernoff bounds we prove that the random variable X_i corresponding to the number of parties from $\hat{\mathcal{P}}_i$ selected in the lottery is concentrated around the (expected) value:

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

i.e., for any constant $\lambda_i \in (0,1)$:

$$\Pr\left[|(1-\lambda_i)x_i \leq X_i \leq (1+\lambda_i)x_i\right] \geq 1 - \mu_i(k), \tag{2}$$

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

[15] 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.

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

1. Divide the reputation parties into m tiers $\hat{\mathcal{P}}_1, \ldots, \hat{\mathcal{P}}_m$ as follows,[a]: For $i = 0 \ldots, m-1$, define $\hat{\mathcal{P}}_{m-i}$ to be the set of parties in $\hat{\mathcal{P}}$ with reputation $\mathtt{Rep}_j \in (\frac{i}{m} + \delta, \frac{i+1}{m} + \delta]$.
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 $i = 1, \ldots, m-1$: Let
$$\ell_i = \frac{\sum_{j=1}^{i} \alpha_j}{\sum_{j=1}^{m} \prod_{q=j}^{m} c_q} \frac{\alpha_{i+1} \prod_{j=i}^{m} c_j - \alpha_i \prod_{j=i+1}^{m} c_j}{\alpha_{i+1}\alpha_i} \log^{1+\epsilon} n$$
and let
$$\ell_m := \frac{\sum_{j=1}^{m} \alpha_j}{\sum_{j=1}^{m} \prod_{q=j}^{m} c_q} \frac{1}{\alpha_m} \log^{1+\epsilon} n$$
5. For each $i = 1, \ldots, m-1$ do the following:
   - If $|\cup_{j=1}^{i} \hat{\mathcal{P}}_j| \geq \ell_i$:
     (a) 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 $\cup_{j=1}^{i} \hat{\mathcal{P}}_j$.
     (b) For each $j \in [i]$ compute $\mathcal{Q}_{i,j}^{\text{col}} := \mathcal{Q}_i \cap \bar{\mathcal{P}}_j$ and update $\bar{\mathcal{P}}_j := \bar{\mathcal{P}}_j \cup (\mathcal{Q}_i \cap \hat{\mathcal{P}}_j)$.
     (c) For each $j \in [i]$ if $\mathcal{Q}_{i,j}^{\text{col}} \neq \emptyset$, then
       i. 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 $\mathtt{A}_{\max}$.
       ii. 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 $\bar{\mathcal{P}}_j := \bar{\mathcal{P}}_j \cup \mathcal{Q}_{i,j}^{+}$.
       iii. Set $\mathcal{Q}_i^{+} := \cup_{j=1}^{m} \mathcal{Q}_{i,j}^{+}$
   - 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 $\log^{1+\epsilon} n$.
6. 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.

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

Hence, by inspection of the protocol one can verify that the x_i's and the ℓ_j's satisfy the following system of linear equations:

$$(x_1, \ldots, x_m)^T = B \cdot (\ell_1, \ldots, \ell_m)^T \tag{3}$$

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

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

The above system of m equations has $2m$ unknowns. To solve it we add the following m equations which are derived from the desired reputation fairness: For each $i \in [m-1]$:

$$x_i := c_i \cdot x_{i+1} \tag{4}$$

and

$$\sum_{i=1}^{m} x_i = \log^{1+\epsilon} k \tag{5}$$

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

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

for each $i \in [m-1]$, and

$$\ell_m := \frac{\sum_{j=1}^{m} \alpha_j}{\sum_{j=1}^{m} \prod_{q=j}^{m} c_q} \frac{1}{\alpha_m} \log^{1+\epsilon} n.$$

This already explains some of the mystery around the seemingly 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

$$\sum_{j=1}^{m} \ell_j = \sum_{i=1}^{m} x_i \overset{Eq.\ 5}{=} \log^{1+\epsilon} k \tag{6}$$

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

Property 2 is proven 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 $\mathsf{RandSet}$ (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 the adversary corrupts for free all the selected parties from the lowest half reputation tiers, still the upper half will have a strong super-majority to compensate so that overall the majority is honest.

Finally, the c-fairness (Property 3) is argued as follows:

–The c-Representation Fairness follows directly from Equations 1, 2 and 4.

–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 $c' < c$: $\frac{p_i}{p_{i+1}} \geq c'$.

In Case 2 the lottery is reset and the output $\mathcal{P}_{\mathtt{sel}}$ is selected by means of invocation of algorithm $\mathtt{A}_{\max}$. This is the simpler 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 $\mathtt{A}_{\max}$ is only invoked if a reset occurs, i.e., if in some step there are no sufficiently many parties to select from; this occurs only if any every set $\hat{\mathcal{P}}_i$ does not have sufficiently many parties to choose from. But the above analysis, for $\delta < \gamma - 1$, the sampling algorithms choose at most $(1+\delta)\log^{1+\epsilon} n$ with overwhelming probability. Hence when each $\hat{\mathcal{P}}_i$ has size at least $\gamma \cdot \log^{1+\epsilon} n$, with overwhelming probability no reset occurs. In this case, by inspection of the protocol one can verify that the number of selected parties is $|\mathcal{P}_{\mathtt{sel}}| = \log^{1+\epsilon} n$.

4 The PoR-Blockchain

We next describe how a PoR-fair lottery can be used to obtain a PoR-blockchain. 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. 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 abstract this by means of a predicate $\mathsf{Validate}$, 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 for proposing and agreeing on the block of any given slot is as follows: A small (i.e., polylogarithmic) slot-committee $\mathcal{C}_{\mathrm{BA}}$ is chosen using our above lottery—recall that the lottery will guarantee that the majority in $\mathcal{C}_{\mathrm{BA}}$ is honest and therefore it can run Byzantine agreement protocols (Consensus and Broadcast). From $\mathcal{C}_{\mathrm{BA}}$ a smaller (constant-size) committee $\mathcal{C}_{\mathrm{BC}}$ is randomly chosen to broadcast its

transactions to everyone. 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 broadcasted by means of protocol Broadcast. We will use *multicasting* to refer to the process of sending a message through the diffusion network to all parties.

Using Broadcast to communicate the transactions known to $\mathcal{C}_{\text{BC}}$ allows us to agree on the union of all transactions known to these parties. However, broadcasting to the whole player set has a big communication and round overhead. To avoid the overhead we use the following idea: Recall that the parties in $\mathcal{C}_{\text{BA}}$ are all reputation parties and can therefore communicated directly. Thus, instead of directly broadcasting to the whole party set $\mathcal{P}$ the parties in $\mathcal{C}_{\text{BC}}$ broadcast to the parties in $\mathcal{C}_{\text{BA}}$. [16] The security of the broadcast protocol ensures that the parties in $\mathcal{C}_{\text{BA}}$ agree on the broadcasted messages and therefore also on the union. What remains is to extend this agreement to the whole player set. This can be easily done since the majority of the parties in $\mathcal{C}_{\text{BA}}$ is honest: Every party in $\mathcal{C}_{\text{BA}}$ signs the union of the received broadcasted messages and sends it to every party in $\mathcal{P}$. The fact that $\mathcal{C}_{\text{BA}}$ has honest majority implies that the only message that might be accepted is this agreed upon union. Hence, once any $P \in \mathcal{P}$ receives such a majority-supported set, he can adopt it as the contents of this slot's block. The above approach brings an asymptotic reduction on the communication complexity of the protocol from $\Omega(n^2)$ down to $O(n \log^{\epsilon} n)$, for some constant $\epsilon > 1$. (The worst-case round complexity also has an asymptotic reduction but this depends on the actual reputation system and the choice of protocol Broadcast.) Additionally, the fact that reputation parties communicate over point-to-point (rather than gossiping) network is likely to further to improve the concrete communication complexity, at least for certain network topologies.

A remaining question is: Where does the randomness for the selection of $\mathcal{C}_{\text{BA}}$ and $\mathcal{C}_{\text{BC}}$ come from? For the static reputation-restricted adversary considered here, we extract the randomness for choosing each $\mathcal{C}_{\text{BA}}$ by repeated calls to the random oracle. In the appendix we discuss how we can extend our treatment using standard techniques to tolerate epoch-resettable adversaries. The formal description of the protocol for announcing and agreeing on the next block can be found in Figure 2. The proof of the following theorem follows the above line of argument and can also be found in Appendix D.2.

$\mathsf{BlockAnnounce}(\hat{\mathcal{P}}, \mathcal{P}, \mathsf{Rep}, B_{\rho-1}, \rho, \delta, \epsilon, L = O(1))$:

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

[a] As is common in blockchains assuming a compact representation of the block, e.g., a Merkle-tree will allow for more efficiency by signing just the root.

Fig. 2: Block announcing protocol for Slot ρ

[16] For clarity in our description we will use a deterministic broadcast protocol for Broadcast, e.g., the Dolev-Strong broadcast protocol [16] for which we know the exact number of rounds. However, since our lottery will ensure honest majority in $\mathcal{C}_{\text{BA}}$, using the techniques by Cohen *et al.* [12,13], we can replace the round-expensive Dolev-Strong broadcast protocol by an randomized, expected-constant round broadcast protocol for honest majorities, e.g., [27].

Theorem 2. *Let* Rep *be a reputation system,* ϵ *and* δ *be small positive constants,* L *denote our sampling algorithm (lottery) discussed in the previous section used for choosing* $\mathcal{C}_{BA}$ *according to* Rep*, and the* c_i*'s be as in Theorem 1. If for a static adversary* $\mathcal{A}$*,* Rep *is* ϵ*-feasible, and all parties in* $\hat{\mathcal{P}}_\rho$ *participate in slot* ρ *then in protocol* 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 BlockAnnounce it is straightforward to build a blockchain-based ledger: In each round parties collect all valid transactions they know and execute BlockAnnounce. For completeness, we provide a description of this simple reputation-based blockchain ledger protocol in Appendix B. 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, to dynamic reputation systems with epoch-resettable adversaries. This extension will also extend the corresponding PoR-based blockchain protocol to that setting.

5 The PoR/PoS-Hybrid Blockchain

As discussed in the introduction, a purely PoR-based blockchain protocol can be insecure if the reputation system is inaccurate. To fortify our ledger against such a situation we do the following: In parallel to the PoR-based blockchain above, we run an independent (and potentially slower) Nakamoto-style ledger protocol. As discussed, we focus our description on Proof-of-Stake-based blockchain (in short, PoS-blockchain) but our treatment can be applied to proof-of-work or even iterated-BFT ledger protocols [4]. As long as the majority of the stake is in honest hands the *back-up PoS-blockchain* will ensure that an inaccurate (or manipulated) reputation system does not result in an insecure ledger.

More concretely, our PoR/PoS-hybrid design ensures the following: If the reputation system is accurate, i.e., it reflect the actual probabilities that the parties behave honestly, then our protocol will be secure and achieve the claimed efficiency. If the reputation is inaccurate and, as a result, a fork is created, but the honest majority of stake assumption holds, then parties will be able to detect the fork—and agree on this fact—within a small number of rounds. We stress that our design uses the assumptions in a tired manner: as long as the reputation system is accurate, the backup blockchain can neither yield false detection positives nor slow down the progress of the PoR-blockchain, even if the majority stake in the system has shifted to malicious parties (in which case the back-up PoS-blockchain might fork). However, if both accuracy of reputation and honest majority of stake fail, the security of the system cannot be guaranteed as with any construction based on assumptions.

Here is how our hybrid ledger works: In every round the reputation parties post to 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 she can complain by posting the block in his local PoR-chain, which is inconsistent with the pointer seen on the backup PoS-blockchain, along with the supporting set of signatures. 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 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. In either case, any such accusation will result in detecting a malicious party: either the accuser, or the accused or some party that issued conflicting signatures on the PoR blockchain. (The detected party's reputation will be then reduced to 0 and the party will be excluded from the protocol.) The detailed specification of our PoR/PoS-hybrid protocol $\Pi^{BC}_{\text{PoR/PoS}}$ is provided in Appendix C along with the proof of the following theorem, where we say that the reputation system is *accurate* if it reflects, within a negligible error the actual probabilities that the reputation parties are honest:

Theorem 3. *Let,* ϵ, δ, c *and* L *be as in Theorem 2. 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 Finality):* *At the end of every block-slot, all reputation parties have the same view of the blockchain.*
– *Block Liveness:* *A new block is added to each local chain held by any reputation party at the end of each 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.

Note that since all honest (non-reputation) parties are connected with the reputation parties via a diffusion network, all that above guarantees will also be offered to them with a delay equal to the maximum delay for any message from a reputation party to reach a non-reputation party in the network.

Detecting Liveness Attacks The above design detects safety violations, i.e., forks, but does not detect the following attack on liveness: A flawed reputation system might allow the attacker controlling a majority of slot committee members to prevent the blockchain from advancing, by not issuing a block signed with sufficiently many signatures. Nonetheless, a mild additional assumption on the accuracy of the reputation system, i.e., that within a polylogarithmic number of rounds an honest-majority committee will be elected, allows to break this deadlock and detect such an attack. In a nutshell, to make sure the above attack to liveness is exposed we employ the following mechanism: In every round, if a reputation party does not receive any block with majority support from the current committee in the last round of BlockAnnounce, then it issues a complaint on the fallback chain. If for any upcoming round ρ there have been complaints posted on the main chain by a majority of the members of $\mathcal{C}_{\text{BA}}$ corresponding to round ρ, then the parties decide that the PoR-blockchain has halted. This ensure that at the latest when the next honest majority committee is elected, the above liveness attack will be exposed. We refer to Appendix C.2 for a more detailed discussion.

6 Conclusions and Open Problems

Reputation has the potential to yield a highly scalable decentralized ledger. However, one needs to be careful in using it as it is a manipulable and subjective resource. We put forth and proved the security of a hybrid design which enjoys efficiency and scalability benefits by using reputation, while fortifying its security with a fallback blockchain relying on standard assumption such as honest majority of stake. Central in our treatment is a new (reputation-)fairness notion which aims to facilitate inclusivity of the resulting system. Our work establishes the basic security principles and the core distributed protocol of such a fair PoR/PoS-hybrid blockchain ledger. We believe that our treatment will seed a further investigation of reputation as a venue for scalable decentralization. To this direction, in addition to the various extensions pointed to throughout the paper and discussed in the appendix, there are two important research directions: (1) A rational analysis and associated mechanism that add economic robustness to the arguments and demonstrate the decentralization forces, and (2) a reliable mechanism for assigning reputation of the parties, e.g. using AI, and adjusting it according to their behavior both in the protocol, as well as potentially on the external recommendation systems.

Acknowledgements. This research was supported by Sunday Group, Inc. The authors would like to thank Yehuda Afek for useful discussions.

References

1. 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.
2. 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.

3. 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.
4. Christian Badertscher, Peter Gazi, Aggelos Kiayias, Alexander Russell, and Vassilis Zikas. Consensus redux: Distributed ledgers in the face of adversarial supremacy. Cryptology ePrint Archive, Report 2020/1021, 2020. `https://eprint.iacr.org/2020/1021`.
5. 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.
6. 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.
7. 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.
8. 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`.
9. Vitalik Buterin. A next-generation smart contract and decentralized application platform, 2013. `https://github.com/ethereum/wiki/wiki/White-Paper`.
10. Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.
17. 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.
18. 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.
19. 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.
20. 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.
21. 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.
22. 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`.
23. 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.
24. 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.
25. Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.
26. 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.

27. 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.
28. 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.
29. 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.
30. 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.
31. 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.
32. 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.
33. 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.
34. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. `http://bitcoin.org/bitcoin.pdf`.
35. 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.
36. Rafail Ostrovsky and Moti Yung. How to withstand mobile virus attacks (extended abstract). In Luigi Logrippo, editor, *10th ACM PODC*, pages 51–59. ACM, August 1991.
37. 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.
38. 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.
39. 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.
40. 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 Supplementary Material to Section 3

A.1 A Simple PoR-Fair Lottery for a 2-Tier Reputation System

For completeness, we first repeat the informal goal of PoR-fairness in our 2-tier example.

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. $x_1 = 2\max\{1, \frac{\alpha_1}{\alpha_2}\}x_2$.
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. Let $c := 2\max\{1, \frac{\alpha_1}{\alpha_2}\}$. The lottery proceeds as follows:

1. First, we choose $\ell_I = \frac{c\alpha_2 - \alpha_1}{(c+1)\alpha_2} y$ parties randomly from $\hat{\mathcal{P}}_1$;
2. Subsequently, we choose $\ell_{II} := \frac{\alpha_1 + \alpha_2}{(c+1)\alpha_2} y$ parties randomly from the union $\hat{\mathcal{P}}_1 \cup \hat{\mathcal{P}}_2$. If a party from $\hat{\mathcal{P}}_1$ is chosen twice, i.e., in both of the above steps, then choose another party randomly from the parties in $\hat{\mathcal{P}}_1$ 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 third property follows from the fact that the parties are chosen randomly from their respective sets. The following theorem formalizes and proves the fact that, "on average", our above lottery satisfies the first two properties discussed in the above goal.

Theorem 4 (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. $\mathbb{E}(|X_1|) = c\,\mathbb{E}(|X_2|)$

Proof (sketch). We prove the two equations in the theorem separately.

1. $\ell_I + \ell_{II} = (\frac{c\alpha_2 - \alpha_1}{(c+1)\alpha_2} + \frac{\alpha_1 + \alpha_2}{(c+1)\alpha_2})y = y$
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 $\hat{\mathcal{P}}_2$:

$$\mathbb{E}(|L_I \cap \hat{\mathcal{P}}_2|) = 0 \tag{7}$$

and

$$\mathbb{E}(|L_I \cap \hat{\mathcal{P}}_1|) = \ell_I = \frac{c\alpha_2 - \alpha_1}{(c+1)\alpha_2} y \tag{8}$$

In the second phase, there are $\alpha_1 + \alpha_2$ parties to choose from in total, out of which α_2 are in $\hat{\mathcal{P}}_2$ and α_1 are in $\hat{\mathcal{P}}_1$. Since we chose $\frac{y}{1+\gamma}$ parties randomly we have:

$$\mathbb{E}(|L_{II} \cap \hat{\mathcal{P}}_2|) = \frac{\alpha_2}{\alpha_1 + \alpha_2} \cdot \frac{\alpha_1 + \alpha_2}{(c+1)\alpha_2} y = \frac{1}{c+1} y \tag{9}$$

and

$$\mathbb{E}(|L_{II} \cap \hat{\mathcal{P}}_1|) = \frac{\alpha_1}{\alpha_1 + \alpha_2} \cdot \frac{\alpha_1 + \alpha_2}{(c+1)\alpha_2} y = \frac{\alpha_1}{\alpha_2(c+1)} y \tag{10}$$

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

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

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,

$$\begin{aligned}
\mathbb{E}(|X_1|) &= \mathbb{E}(|L \cap \hat{\mathcal{P}}_1|) \\
&= \mathbb{E}(|L_I \cap \hat{\mathcal{P}}_1| + |L_{II} \cap \hat{\mathcal{P}}_1|) \\
&= \mathbb{E}(|L_I \cap \hat{\mathcal{P}}_1|) + \mathbb{E}(|L_{II} \cap \hat{\mathcal{P}}_1|) \\
&= \frac{c\alpha_2 - \alpha_1}{(c+1)\alpha_2} y + \frac{\alpha_1}{\alpha_2(c+1)} y \\
&= \frac{c}{c+1} y
\end{aligned}$$

The above imply:

$$\frac{\mathbb{E}(|X_1|)}{\mathbb{E}(|X_2|)} = c$$

A.2 PoR-Fair Definition and Lottery (Cont'd)

This section includes supplementary material for Section 3.1.

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

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 $T(s_i) \geq \log \frac{|\mathcal{P}|}{t}$, then update $\bar{\mathcal{P}} := \bar{\mathcal{P}} \cup \{P_i\}$.
3. Output $\bar{\mathcal{P}}$.

Fig. 3: 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.

Fig. 4: A simple uniform sampler without replacements based on a hash function h

B Supplementary Material to Section 4

Figure 5 describes our PoR-Blockchain Protocol for static reputation reputation

$\mathsf{RepBC}(\hat{\mathcal{P}}, \mathcal{P}, \mathtt{Rep}, B_{\rho-1}, \rho, \delta, \epsilon, L = O(1))$
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 [12] we can extend our treatment to use randomized—and more round-efficient—protocols.

Fig. 5: The Reputation-based Blockchain (Static Reputation)

C Supplementary material to Section 5

Figure 6 describes our PoR/PoS-Hybrid Blockchain Protocol for static reputation reputation

C.1 Proof of Theorem 3

Proof (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

$\Pi^{BC}_{\text{PoR/PoS}}(\hat{\mathcal{P}}, \mathcal{P}, \texttt{Rep}, B_{\rho-1}, \rho, \delta, \epsilon, L = O(1))$

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 [12] 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

Fig. 6: The PoR/PoS-Hybrid Blockchain Protocol (Static Reputation)

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 [14,3] 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 [21,3]—this happens within in at most $2k$ blocks from round q [3]—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.2 Countering Liveness Attacks

Our safety-violation detection mechanism does not prevent the following attack on liveness in the fallback case: A flawed reputation system might allow the attacker controlling a majority of round committees to prevent the blockchain from advancing, by not issuing a block signed with sufficiently many signatures.

We remark that in order to attack liveness as above, the adversary will need to allow *no* honest reputation party to learn the contents of the current block, as otherwise that honest party will diffuse it to their network and therefore, (1) every reputation party will learn it within one round—since they are connected by a direct channels— and (2) every party will adopt it in time $\Delta_{max} + \Delta_{round}$, where Δ_{max} is the maximum delay for any message from a reputation party to reach a non-reputation party in the network and Δ_{round} is the round duration/time-out.

To make sure the above attack to liveness is exposed we employ the following mechanism: In every round, if a reputation party does not receive any block with majority support from the current committee in the last round of BlockAnnounce, then it issues a complaint on the fallback chain. If for any upcoming round ρ there have been complaints posted on the main chain by a majority in ρ, then the parties detect the halt (and, as above, an external protocol for recalibrating reputations can be employed).

The above mechanism ensures that under a very weak assumption on the accuracy of the reputation system, i.e., that an honest-majority committee is elected within a polylogarithmic number of rounds, any such halt will be detected by the system. Note that as long as the reputation is accurate, the fallback chain will never induce such a detection, as all committees will have honest majority and therefore no honest party complains. Note, also, that our lottery will satisfy the above assumption under very realistic assumptions on the the reputation, e.g., that a constant fraction of the reputations in each of the the high tiers is correct. We refer to the full version of this work for a detailed treatment.

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 suffices to ensure that the system does not preserve deep forks, which in case of cryptocurrencies can lead to double spending. Additionally, we can even add support for recovery, not just detection, 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 H.3 to restart the PoR-blockchain from the point before the detected misbehavior.

Notwithstanding, the above mechanism does not prevent the following attack on liveness: A flawed reputation system might allow the attacker controlling a majority of slot committee members to prevent the blockchain from advancing, by not issuing a block signed with sufficiently many signatures. Nonetheless, a very mild assumption on the accuracy reputation allows to break this deadlock and detect such an attack. In a nutshell, to make sure the above attack to liveness is exposed we employ the following mechanism: In every round, if a reputation party does not receive any block with majority support from the current committee in the last round of BlockAnnounce, then it issues a complaint on the fallback chain. If for any upcoming round ρ there have been complaints posted on the main chain by a majority of the members in ρ, then the parties detect the halt (and, as above, an external protocol for recalibrating reputations can be employed). This ensure that at the latest when the next honest majority committee is elected, the above liveness attack will be exposed.

D Deferred Proofs

D.1 Proof of Theorem 1

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 noes not reset, and Case 2: L resets.

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. Note that $\mathtt{A}_{\max}$ is only invoked if a reset occurs, i.e., if in some step there are no sufficiently

many parties to select from; this never occurs if every set $\hat{\mathcal{P}}_i$ has at least $\gamma \cdot \log^{1+\epsilon} n$ and, therefore, c-fairness does not apply in this case.

In Case 1, The lottery is never reset. This case is the bulk of the proof. 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.

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

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:

$$x_{i,j} := Exp(X_{i,j}) = \alpha_i \cdot \frac{\ell_j}{\sum_{q=1}^{j} \alpha_q} \tag{11}$$

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

$$\Pr\left[|(1-\lambda_{i,j})x_{i,j} \leq X_{i,j} \leq (1+\lambda_{i,j})x_{i,j}\right] \geq 1 - \mu_{i,j}(k), \tag{12}$$

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 11 implies that

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

and Equation 12 implies that for any $(\lambda_{i,1}, \ldots, \lambda_{i,m}) \in (0,1)^m$

$$\Pr\left[|(1-\sum_{j=1}^{m} \lambda_{i,j})x_{i,j} \leq \sum_{j=1}^{m} X_{i,j} \leq (1+\sum_{j=1}^{m} \lambda_{i,j})x_{i,j}\right] \geq 1 - \mu_i(k), \tag{13}$$

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:

$$(x_1, \ldots, x_m)^T = B \cdot (\ell_1, \ldots, \ell_m)^T \tag{14}$$

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

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

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

– For each $i \in [m-1]$:

$$x_i := c_i \cdot x_{i+1} \tag{15}$$

–

$$\sum_{i=1}^{m} x_i = \log^{1+\epsilon} k \tag{16}$$

This yields 2m linear equations. By solving the above system of equations we can compute:
For each $i = 1, \ldots, m-1$:

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

and

$$\ell_m := \frac{\sum_{j=1}^{m} \alpha_j}{\sum_{j=1}^{m} \prod_{q=1}^{j} c_q} \frac{1}{\alpha_m} \log^{1+\epsilon} n$$

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

$$\sum_{j=1}^{m} \ell_j = \sum_{i=1}^{m} x_i \overset{Eq.\ 16}{=} \log^{1+\epsilon} k \tag{17}$$

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

$$(\ell_1, \ldots, \ell_m)^T = B^{-1}(x_1, \ldots, x_m)^T \tag{18}$$

We are now ready to prove properties 1 and 2 in the theorem. We start with Property 1:

Claim. 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 $j \in [m]$:

$$Exp(|L_j|) = \ell_j \tag{19}$$

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

$$\Pr\left[|(1-\zeta_j)\ell_j \leq L_j \leq (1+\zeta_j)\ell_j\right] \geq 1 - \hat{\mu}_j(k), \tag{20}$$

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 19 we have

$$Exp(|\mathcal{P}_{\mathtt{sel}}|) = \sum_{j=1}^{m} \ell_j \overset{Eq.17}{=} log^{1+\epsilon} k$$

Similarly, from Equation 20 we get

$$\Pr\left[|(1-\sum_{j=1}^{m}\zeta_j)\sum_{j=1}^{m}\ell_j \leq \sum_{j=1}^{m}L_j \leq (1+\sum_{j=1}^{m}\zeta_j)\sum_{j=1}^{m}\ell_j\right] \geq 1-\hat{\mu}(k), \tag{21}$$

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

$$\Pr\left[|(1-\zeta)\log^{1+\epsilon} k \leq |P_{\mathtt{sel}}| \leq (1+\zeta)\log^{1+\epsilon} k\right] \geq 1-\hat{\mu}(k), \tag{22}$$

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

We next proceed to Property 2:

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. 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 20 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 $|\mathcal{Q}^{\mathrm{col}}_{i,j}| < \delta'''|\mathcal{Q}_{i,j}|$. Hence, even if we allow the adversary $\mathcal{A}'$ to corrupt all parties in $\mathcal{Q}^{\mathrm{col}}_{i,j}$, 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 $\mathcal{Q}^{\mathrm{col}}_{i,j} = \omega(|\mathcal{Q}_{i,j}|)$, this implies that $|\mathcal{Q}^+_{i,j}| = O(|Q_i|)$. Hence Equation 20 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 [25] (see Appendix E), 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 $\mathcal{Q}_{i,j}^{\text{col}} = \omega(|\mathcal{Q}_{i,j}|)$, 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, $\mathcal{Q}_{i,j}^{\text{col}} = \Theta(|\mathcal{Q}_{i,j}|)$ 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 12 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 $\mu'_{i,j}$:

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

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

$$\Pr[\forall j:\ C_{i,j} < (\frac{m-i}{m} - \delta')x_{i,j}] \geq 1 - \tilde{\mu}_i(k). \tag{24}$$

which implies that

$$\Pr[\sum_{j=1}^{m} C_{i,j} < \sum_{j=1}^{m}(\frac{m-i}{m} - \delta')x_{i,j}] \geq 1 - \mu''_i(k), \tag{25}$$

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 25 can be rewritten as follows

$$\Pr[C_i < (\frac{m-i}{m} - \delta')x_i] \geq 1 - \mu''_i(k) \tag{26}$$

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 26, 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:

$$\begin{aligned} corr_1 &= C_1 + \frac{m-2}{2m}x_1 \\ &< (\frac{1}{m} + \frac{m-2}{2m} - \delta')x_1 \\ &= (1/2 - \delta')x_1 \end{aligned} \tag{27}$$

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

$$R < \frac{(1/2 - \delta')x_1}{x_1 + \sum_{i=m/2}^{m} x_i} < 1/2 - \delta' \tag{28}$$

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 lottery 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 15.
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 $i = 1, \ldots, m-1$:

$$\begin{aligned} \frac{p_i}{p_{i+1}} &\geq \frac{(1 - \lambda_i)x_i}{(1 + \lambda_{i+1})x_{i+1}} \cdot \frac{\alpha_{i+1}}{\alpha_i} \\ &= \frac{(1 - \lambda_i)}{(1 + \lambda_{i+1})} \cdot \frac{x_i}{x_{i+1}} \cdot \frac{\alpha_{i+1}}{\alpha_i} \\ &= \frac{(1 - \lambda_i)}{(1 + \lambda_{i+1})} \cdot c_i \cdot \frac{\alpha_{i+1}}{\alpha_i} \\ &\geq \frac{(1 - \lambda_i)}{(1 + \lambda_{i+1})} c \end{aligned} \tag{29}$$

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 $\frac{p_i}{p_{i+1}} \geq c'$.

In Case 2 the lottery is reset and the output $\mathcal{P}_{\mathtt{sel}}$ is selected by means of invocation of algorithm $\mathtt{A}_{\max}$. This is the simpler 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 $\mathtt{A}_{\max}$ is only invoked if a reset occurs, i.e., if in some step there are no sufficiently many parties to select from; this occurs only if any every set $\hat{\mathcal{P}}_i$ does not have sufficiently many parties to choose from. But the above analysis, for $\delta < \gamma - 1$, the sampling algorithms choose at most $(1 + \delta) \log^{1+\epsilon} n$ with overwhelming probability. Hence when each $\hat{\mathcal{P}}_i$ has size at least $\gamma \cdot \log^{1+\epsilon} n$, with overwhelming probability no reset occurs. In this case, by inspection of the protocol one can verify that the number of selected parties is $|\mathcal{P}_{\mathtt{sel}}| = \log^{1+\epsilon} n$.

D.2 Proof of Theorem 2

Proof (sketch). 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) along with the randomness that seeds the lottery. Note that in the static adversary considered here, this randomness is independent from the randomness that samples the corrupted set. Since $\mathcal{C}_{\mathrm{BA}}$ is selected by means of L, and the lottery is ϵ-feasible, Theorem 1 ensures that the majority of the parties in $\mathcal{C}_{\mathrm{BA}}$ is honest. This means that we can use a Byzantine broadcast protocol to have any party $\hat{P}_i \in \mathcal{C}_{\mathrm{BA}}$ consistently broadcast any messages to all the parties in $\mathcal{C}_{\mathrm{BA}}$, 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 $Y = \hat{T}$ 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.

E Hoeffdings Inequality

Lemma 5. (Hoeffding's Inequality [25]) *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 $\delta > 0$, $\Pr[\overline{X} - \mu \geq \delta] \leq e^{-\frac{2n\delta^2}{(b-a)^2}}$.*

F Reputation Systems with Limited Correlations

Our treatment discusses a wide yet restricted 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 [20].

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 [8], 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 [1, 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* $\mathtt{Rep}$ *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* $\mathtt{Rep}$*, that* $\hat{P}_i$ *gets corrupted. Let* $\mathtt{Rep}_{IND}$ *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* $\mathtt{Rep}$ *is* δ*-correlation-free reputation system if and only if the statistical distance of* $\mathtt{Rep}$ *and* $\mathtt{Rep}_{IND}$ *is at most* δ*.*

It is straightforward to verify that is δ is very small (negligible in $|\hat{\mathcal{P}}|$) and $\mathtt{Rep}_{IND}$ is feasible, then our sampling algorithm discussed above will directly select sets with honest majority. Furthermore, assuming sufficiently many parties with high reputation in $\mathtt{Rep}_{IND}$, 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.

G Dynamic Reputation and Epoch-Resettable Adversaries

We can also extend our analysis for announcing the blocks in each slot, and thereby our ledger protocol, to cope with epoch-resettable adversaries. In fact, as long as the reputation system ensemble is a sequence of independent (static) reputation systems, we do not need to make any modification to the protocol other than requiring the parties to 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 [6,26]. Indeed, although the epoch-resettable adversary will able to predict the next epoch's committees, he is still bound to select his corrupted parties independently according to their (new) reputation. As long as all reputation vectors (i.e., for each epoch) are feasible, such an adversary will be unable to have a corrupted majority committee chosen, except with negligible probability.

However the above assumption that reputation-vectors of different epoch are independent, might fall short in capturing situations, where malicious parties might try to manipulate their reputation (e.g., by temporarily playing honestly) depending on the protocol's history. Under the assumption that reputation for the next epoch is already define in the one-but-last slot of the current epoch[17] we can tolerate an epoch-resettable adversary by modifying the way this randomness is selected as follows: We adapt the block announcing protocol to have every party in any committee include, in addition to his transaction pool, a fresh random nonce, and the concatenation of all these nonces is included to the block along with the union of the transactions posted by $\mathcal{C}_{\mathrm{BC}}$. (The corresponding protocol is given in Figure 7). As the distribution of the reputation vector is decided before the last slot, the choice of all parties in the next epoch will be independent of this distribution and therefore the same guarantees as before will apply.

Unfortunately, the above idea is still problematic. Indeed, since with an epoch resettable adversary the corrupted set is "reset" at the beginning of each epoch, the adversary across two epochs might be able to take keys he learned in the first of the two epoch and use them at the beginning slot of the second epoch (before parties have had a chance to erase) thereby effectively holding 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:

[17] This is a natural assumption as any reputation adjusting mechanism will need to have parties agree on the adjustment of the reputation, e.g., by extracting is from blocks deep enough in the blockchain.

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

1. Extract the concatenation of randomness r_{-1} from the blocks of the previous epoch.
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 1, 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 $\mathcal{P}$.
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.

Fig. 7: Block announcing protocol for Slot ρ for an epoch resettable adversary

Definition 5. *For a reputation system* $\mathtt{Rep}$ *for parties from a reputation set* $\hat{\mathcal{P}}$*, a (possibly probabilistic) algorithm* $\mathtt{A}$ *for sampling a subset of parties from* $\hat{\mathcal{P}}$*, and an* $\mathtt{Rep}$*-adversary* $\mathcal{A}$*, we say that* $\mathtt{Rep}$ is strongly $(\epsilon, \mathtt{A})$-feasible for $\mathcal{A}$ *if, with overwhelming probability,*[18] $\mathtt{A}$ *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 $\mathtt{Rep}$ from the entire reputation-party set $\hat{\mathcal{P}}$, and independently of the coins of $\mathtt{A}$. (Indeed, otherwise it would be trivial to always corrupt a majority.)

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

H Extensions and Future Research

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

Our hybrid protocol $\Pi^{BC}_{\mathrm{PoR/PoS}}$ 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

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

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 [29]:

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.

H.2 Relaxing Synchrony

Removing the arguably strong simplifying assumptions of perfect synchrony with zero-delay channels is another interesting direction. We conjecture one can adapt techniques from the blockchain literature, e.g., [12,13] for removing the deterministic zero-delay assumption and clock synchronizarion techniques synchronization [15,30,24,18,17,2,39,31,35] to relax the perfect synchrony assumption.

H.3 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, $\mathit{\Pi}^{BC}_{\text{PoR/PoS}}$ assumes a very simple incentive structure to characterize the reputation parties' rationality: Reputation parties should always prefer higher 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[19]: Periodically,

[19] 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.

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 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 [14,3,22] 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.

H.4 Dynamically 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.