The Economic Limits of Permissionless Consensus

ERIC BUDISH, University of Chicago, USA
ANDREW LEWIS-PYE, London School of Economics, UK
TIM ROUGHGARDEN, Columbia University & a16z Crypto, USA

An ideal permissionless consensus protocol would, in addition to satisfying standard consistency and liveness guarantees, render consistency violations prohibitively expensive for the attacker without collateral damage to honest participants—for example, by programatically confiscating an attacker's resources without reducing the value of honest participants' resources, as is the intention for slashing in a proof-of-stake protocol. We make this idea precise with our notion of the EAAC (expensive to attack in the absence of collapse) property, and prove the following results:

- In the synchronous and dynamically available setting (in which the communication network is reliable but non-malicious players may be periodically inactive), with an adversary that controls at least one-half of the overall resources, no protocol can be EAAC. In particular, this result rules out EAAC for all typical longest-chain protocols (be they proof-of-work or proof-of-stake).
- In the partially synchronous and quasi-permissionless setting (in which resource-controlling non-malicious players are always active but the communication network may suffer periods of unreliability), with an adversary that controls at least one-third of the overall resources, no protocol can be EAAC. In particular, slashing in a proof-of-stake protocol cannot achieve its intended purpose if message delays cannot be bounded a priori.
- In the synchronous and quasi-permissionless setting, there is a proof-of-stake protocol with slashing that, provided the adversary controls less than two-thirds of the overall stake, satisfies the EAAC property.

Our work formalizes the potential security benefits of proof-of-stake sybil-resistance coupled with slashing and the common belief that the merge has increased Ethereum's economic security. Our work also provides mathematical justifications for several key design decisions behind the post-merge Ethereum protocol, ranging from long cooldown periods for unstaking to economic penalties for inactivity.

CCS Concepts: • **Theory of computation → Theory and algorithms for application domains**; **Algorithmic game theory and mechanism design**;

Additional Key Words and Phrases: Distributed Computing, Consensus Protocols, Blockchain Protocols, Ethereum, Proof-of-Stake, Accountability

ACM Reference Format:
Eric Budish, Andrew Lewis-Pye, and Tim Roughgarden. 2024. The Economic Limits of Permissionless Consensus. In *The 25th ACM Conference on Economics and Computation (EC '24), July 8–11, 2024, New Haven, CT, USA.* ACM, New York, NY, USA, 28 pages. https://doi.org/10.1145/3670865.3673548

Authors' Contact Information: Eric Budish, Booth School of Business, University of Chicago, Chicago, USA, eric.budish@chicagobooth.edu; Andrew Lewis-Pye, Department of Mathematics, London School of Economics, London, UK, a.lewis7@lse.ac.uk; Tim Roughgarden, Computer Science Department, Columbia University & a16z Crypto, New York, USA, tim.roughgarden@gmail.com.

EC '24, July 8–11, 2024, New Haven, CT, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0704-9/24/07
https://doi.org/10.1145/3670865.3673548

1 Introduction

1.1 The Security of Permissionless Consensus Protocols

The core functionality of a blockchain protocol such as Bitcoin or Ethereum is *permissionless consensus*, with a potentially large and ever-evolving set of participants kept in sync on the state of the blockchain via a consensus protocol. Compared to the traditional setting of permissioned consensus protocols (with a fixed and known participant set), permissionless protocols must cope with three novel challenges (cf., [Lewis-Pye and Roughgarden, 2023]):

- **The unknown players challenge.** The set of participants is unknown at the time of protocol deployment and is of unknown size.
- **The player inactivity challenge.** Participants can start or stop running the protocol at any time.
- **The sybil challenge.** One participant may masquerade as many by using many identifiers (a.k.a. "sybils").

Reasoning about the security of a *permissioned* consensus protocol is, to a large extent, a purely computer science question. (Here "security" means that the protocol satisfies both liveness and consistency—as long as there's work to be done it gets done, and without any two participants ever committing to conflicting decisions.) Consider, for example, a corporation that wants to grant customers access to a database while keeping its downtime percentage to 0.0001%. One approach would be to replicate the database on many different servers and use a consensus protocol to keep those replicas in sync. How many replicas are necessary and sufficient to achieve the desired uptime? Or, in two parts:

(Q1) For a given value of k, how many replicas $n(k)$ are necessary and sufficient to guarantee security even when k of the replicas have failed?

(Q2) For a given target downtime percentage δ, what is the smallest value of k such that the probability that at least $k + 1$ of $n(k)$ replicas fail simultaneously is at most δ?

The distributing computing literature resolves question (Q1) for a staggering variety of settings (with $n(k) = 2k+1$ and $n(k) = 3k+1$ being two of the most common answers); see, e.g., [Cachin et al., 2011, Lynch, 1996] for an introduction. Given a probabilistic model of replica failures, question (Q2) then boils down to a calculation, providing the corporation with the appropriate value of k and the corresponding number $n(k)$ of servers that it should buy.

Reasoning about the security of a *permissionless* consensus protocol fundamentally requires the synthesis of computer science and economic arguments. First, the sybil challenge generally forces such a protocol to measure "size" in terms of some resource that is, unlike identifiers, scarce and therefore costly. Common examples include hashrate (as in a proof-of-work protocol) and staked cryptocurrency (as in a proof-of-stake protocol). Second, while misbehaving replicas in a permissioned protocol are generally attributed to hardware failures and software bugs, permissionless protocols must tolerate deliberately malicious behavior by a motivated attacker (a hacker, designers of a competing protocol, or even an unfriendly nation-state). The analogs of questions (Q1) and (Q2) are then:

(Q3) What is the largest value of ρ for which a protocol can guarantee security even when a ρ fraction of the costly resource is controlled by an attacker?

(Q4) How unlikely is it that an attacker controls more than a ρ fraction of the costly resource and then carries out an attack?

Question (Q3) is well defined, and the last decade of research on blockchain protocols has answered it in a range of settings. Question (Q4) makes no sense without an economic model for the cost of carrying out an attack. The goal of this paper is to develop a mathematical framework for quantifying this cost and for designing permissionless consensus protocols in which this cost is as large as possible.

1.2 The Economic Consequences of an Attack: Scorched Earth vs. Targeted Punishment

In the Bitcoin white paper, Nakamoto [Nakamoto et al., 2008] noted that an attacker controlling 51% of the overall hashrate could force consistency violations and thereby carry out double-spend attacks, but suggested that the consequent economic cost might make such an attack unprofitable:

> If a greedy attacker is able to assemble more CPU power than all the honest nodes, he would have to choose between using it to defraud people by stealing back his payments, or using it to generate new coins. He ought to find it more profitable to play by the rules, such rules that favour him with more new coins than everyone else combined, than to undermine the system and the validity of his own wealth.

This argument rests on the assumption that a double-spend attack would cause a significant and permanent drop in the USD-denominated market price of Bitcoin’s native currency BTC, with the attacker then foregoing most of the USD-denominated value of the future BTC block rewards that it’s positioned to receive.

More recently, this initial narrative around the security of Bitcoin and other proof-of-work blockchain protocols has evolved into a second narrative, for two reasons. First, empirical evidence for the “double-spends will crash the cryptocurrency price” hypothesis has been weak. Second, the “CPUs” that Nakamoto referred to have been almost entirely replaced by ASICs that serve no purpose other than to evaluate a hard-coded cryptographic hash function such as SHA-256. Now, the story goes: if an attacker with 51% of the hashrate were to carry out a double-spend attack on (say) the Bitcoin protocol, the Bitcoin ecosystem could respond with the “nuclear option,” changing the cryptographic hash function used for proof-of-work mining via a coordinated upgrade to the protocol (a “hard fork”). Such an upgrade would render existing ASICs useless, leaving the attacker with a defunct pile of scrap metal. Hopefully, the mere threat of this nuclear option would deter any potential attackers, and the option would never have to actually be exercised.

An alarming aspect of both narratives is the “scorched earth” nature of an attack’s consequences: honest participants (passive holders of BTC or ASIC-owning honest miners, respectively) are harmed as much as the attacker. Is scorched earth-style punishment fundamental to permissionless consensus, or an artifact of the specific design decisions made in Bitcoin and other proof-of-work protocols?

Ideally, a blockchain protocol could punish an attacker that carries out a double-spend attack in a targeted and non-scorched-earth way, leaving honest participants unharmed. The hope for such “asymmetric punishment” has long rested with proof-of-stake blockchain protocols, in which the “power” of a participant is proportional to how much of the protocol’s native currency they have locked up in a designated staking contract. Intuitively, with the scarce and costly resource controlled directly by the protocol (rather than “off-chain,” as with hashrate), such a protocol is positioned to directly and surgically confiscate resources from specific participants (perhaps as part of a hard fork, or perhaps programmatically as part of the protocol’s normal operation) [Buterin, 2014b, Kwon, 2014].

The Ethereum protocol, which famously migrated from proof-of-work to proof-of-stake (among many other changes) in September 2022 in an event known as “the merge,” offers an interesting case study. Ethereum’s lead founder, Vitalik Buterin, wrote in the early design stages that “The ‘one-sentence philosophy’ of proof of stake is … ‘security comes from putting up economic value-at-loss’” [Buterin, 2016] and “The intention is to make 51% attacks extremely expensive, so that even a majority of validators working together cannot roll back finalized blocks without undertaking an extremely large economic loss" [Buterin, 2017]. Today, post-merge, the protocol's official documents echo the aspirations above, asserting that "proof-of-stake offers greater crypto-economic security than proof-of-work" and "economic penalties for misbehavior make 51% style attacks more costly for an attacker compared to proof-of-work."[1] What would be a rigorous phrasing of these assertions? Are they true? Could there be alternative protocols—perhaps wildly different from the proof-of-work or proof-of-stake protocols considered to date—that are "still more economically secure" than Ethereum? How do the answers depend on the assumptions imposed on, for example, the reliability of message delivery or the active participation of non-malicious players?

1.3 Overview of Results

Defining the cost of an attack. We augment the permissionless consensus framework of Lewis-Pye and Roughgarden [Lewis-Pye and Roughgarden, 2023] with a new model for reasoning about the cost of causing consistency violations. Informally, we call such an attack *cheap* if the attacker suffers no economic consequences following the consistency violation, and *expensive* otherwise. We call the attack *expensive due to collapse* if the attacker is harmed for the wrong ("scorched earth") reasons, with honest protocol participants also suffering, as in the two narratives around the security of proof-of-work protocols described in Section 1.2. We call the attack *expensive in the absence of collapse* if the attacker is harmed for the right reasons—targeted punishment that avoids collateral damage to the honest participants, as is the goal of slashing in a proof-of-stake protocol. Our key definition is that of an *EAAC* protocol (for "expensive to attack in the absence of collapse") which states, informally, that every attack on the protocol is expensive in the absence of collapse. We then interpret a protocol that guarantees the EAAC property as "more economically secure" than one that does not. We next survey the three main results in this paper (depicted in Figure 1) — two that identify settings in which only "classical security" is possible, and a third showing that, under appropriate assumptions, proof-of-stake protocols with slashing can indeed obtain additional "economic security."

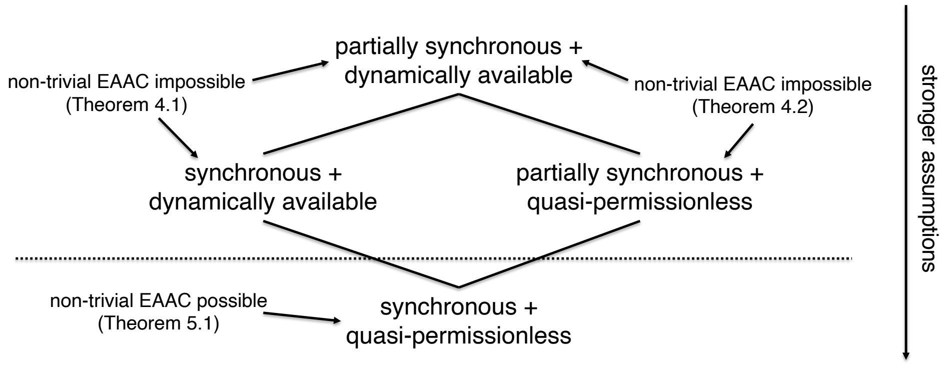

Fig. 1. Summary of main results. While only "classical security" is achievable in the dynamically available setting (in which non-malicious players may be periodically inactive) or the partially synchronous setting (in which the communication network may suffer unbounded periods of unreliability), proof-of-stake protocols with slashing can achieve additional "economic security" in the quasi-permissionless and synchronous setting.

[1]See https://ethereum.org/developers/docs/consensus-mechanisms/pos.

Impossibility of EAAC in the dynamically available setting. Our first main result (Theorem 4.1) states that, in the synchronous and dynamically available setting (in which the communication network is reliable but non-malicious players may be periodically inactive), with an adversary that controls at least one-half of the overall resources, no protocol can be EAAC. This result is optimal with respect to the size of the adversary, as the Bitcoin protocol (among others) guarantees (probabilistic) consistency and liveness in the synchronous and dynamically available setting when the adversary controls less than one-half of the overall hashrate [Garay et al., 2018], and is thus vacuously EAAC in this case (Figure 2). In particular, this result rules out non-trivial EAAC guarantees for all typical longest-chain protocols (be they proof-of-work protocols like Bitcoin or proof-of-stake protocols such as Ouroboros [Kiayias et al., 2017] and Snow White [Daian et al., 2019]).

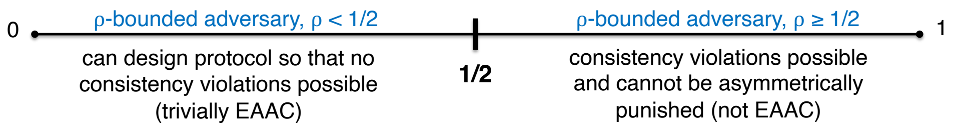
0
ρ-bounded adversary, ρ < 1/2
can design protocol so that no consistency violations possible (trivially EAAC)
1/2
ρ-bounded adversary, ρ ≥ 1/2
consistency violations possible and cannot be asymmetrically punished (not EAAC)
1

Fig. 2. Theorem 4.1. No non-trivial EAAC guarantees are possible in the dynamically available setting, even with synchronous communication: once an adversary is large enough to cause consistency violations, it is also large enough to avoid asymmetric punishment. A ρ-bounded adversary is one that controls at most a ρ fraction of each resource (such as hashrate or stake) used by a protocol.

To give the flavor of the proof, consider two disjoint sets of players X and Y that each own an equal amount of resources. Liveness in the dynamically available setting implies that if one of these sets never hears from the other, it must forge ahead and continue to confirm transactions. So imagine that the players in X are malicious, don't talk to Y, and behave as if they were honest and never heard from Y, confirming transactions to themselves that conflict with the transactions confirmed by Y during the same period. Now suppose that, at some later point, players in X disseminate all the messages that they would have disseminated if honest in their simulated execution. At this point, it is not possible for late-arriving players to determine whether the players in X or the players in Y are honest. If the protocol happens to make this particular attack expensive (by harming the players in X), there is a symmetric execution (with the players in X honest and those in Y malicious) in which the honest players are the ones who are harmed.

Impossibility of EAAC in the partially synchronous setting. Our second result (Theorem 4.2) states that, in the partially synchronous and quasi-permissionless setting (in which resource-controlling non-malicious players are always active but the communication network may suffer periods of unreliability), with an adversary that controls at least one-third of the overall resources, no protocol can be EAAC.[2] (In fact, something stronger is true: no protocol can make consistency violations expensive for the attacker, even allowing for collateral damage to honest players. In particular, slashing in a proof-of-stake protocol cannot achieve its intended purpose if message delays cannot be bounded a priori. This result is optimal with respect to the size of the adversary, as there is a proof-of-stake PBFT-style protocol that guarantees consistency and liveness in the partially synchronous and quasi-permissionless setting when the adversary controls less than one-third of the overall stake [Lewis-Pye and Roughgarden, 2023], which is therefore vacuously EAAC in that regime (Figure 3).

[2]This version of the theorem statement assumes that the protocol is required to be live when the adversary is ρ-bounded for $\rho < 1/3$. The full theorem statement in Section 4 generalises the result to consider weaker liveness requirements.

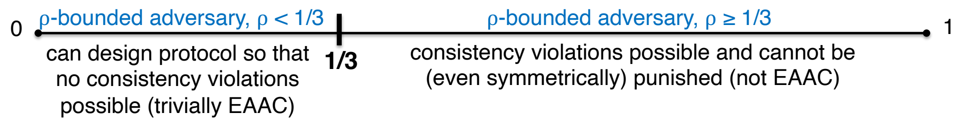
0
ρ-bounded adversary, ρ < 1/3
can design protocol so that no consistency violations possible (trivially EAAC)
1/3
ρ-bounded adversary, ρ ≥ 1/3
consistency violations possible and cannot be (even symmetrically) punished (not EAAC)
1

Fig. 3. Theorem 4.2. No non-trivial EAAC guarantees are possible with partially synchronous communication, even in the quasi-permissionless setting: once an adversary is large enough to cause consistency violations, it is also large enough to avoid (even symmetric) punishment.

To give a sense of the proof of this result, consider three sets of players X, Y, and Z, all with equal resources, with the players in X and Z honest and the players in Y malicious. Suppose that messages disseminated by players in X are received by players in $X \cup Y$ right away but not by players in Z for a very long time (which is a possibility in the partially synchronous model). Symmetrically, suppose Y and Z but not X promptly receive messages sent by players in Z. Suppose further that the malicious players in Y pretend to the players in X that they've never heard from anyone in Z and to the players in Z that they've never heard from anyone in X. Liveness dictates that the players in X and the players in Z must each forge ahead and confirm transactions, even though no messages between players in X and Z have been delivered yet. These uncoordinated confirmed transactions will generally conflict, resulting in a consistency violation. Moreover, this violation may not be noticed by the players of X and Z for a very long time (again due to the arbitrarily long delays in the partially synchronous model), giving the players of Y the opportunity to sell off their resources and avoid any possible punishment in the meantime.

Possibility of EAAC in the synchronous and quasi-permissionless setting. Our final result (Theorem 5.1) states that, in the synchronous and quasi-permissionless setting, there is a proof-of-stake protocol with slashing that, provided the adversary controls less than two-thirds of the overall stake, satisfies the EAAC property. In fact, our protocol is designed for a version of the synchronous setting defined by two known parameters: one parameter Δ that represents typical network speed, perhaps on the order of milliseconds or seconds; and a second parameter Δ^* that represents the time required for players to communicate when the network is unreliable (whether over the network or by out-of-band coordination), perhaps on the order of days or weeks.[3] Intuitively, when there's no attack, the protocol operates at a speed proportional to Δ; under attack, it recovers at a speed proportional to Δ^*.

More precisely, we give a protocol that: (i) is live and consistent in the partially synchronous setting (with respect to the parameter Δ) provided the adversary controls less than one-third of the overall stake; and (ii) is EAAC so long as the adversary controls less than two-thirds of the overall stake and message delays are no larger than Δ^*.[4] A recent result of Tas et al. [Tas et al., 2023], when translated to our model, implies that our result is optimal with respect to the size of the adversary: even in the permissioned and synchronous setting, if a protocol guarantees liveness with respect to an adversary that controls less than one third of the overall resources, it cannot guarantee EAAC with respect to an adversary that controls at least two-thirds of the overall resources (Figure 4).

[3]One interpretation of the parameter Δ^* is as the speed of communication through social channels, as referenced in, for example, Buterin's discussion of"weak subjectivity" [Buterin, 2014a].

[4]This version of the theorem statement assumes that we require protocols to be live and consistent in the partially synchronous setting provided the adversary controls less than one-third of the overall stake. A standard technique (which just modifies the definition of a 'quorum certificate' to require votes from a larger proportion of relevant players) can be used to extend the regime in which our protocol is EAAC beyond the 2/3 bound at the expense of reducing liveness resilience: If the protocol is required to be live for adversaries that are ρ_l-bounded (for some $\rho_l < 1/3$), then it can be made EAAC when the adversary is ρ-bounded for $\rho < 1 - \rho_l$. See Section 9 of the online version of the paper for further details.

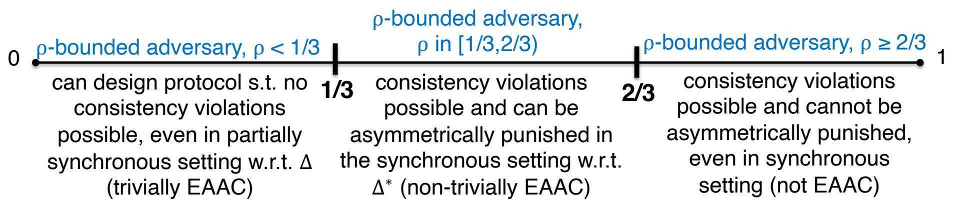
0
ρ-bounded adversary, ρ < 1/3
can design protocol s.t. no consistency violations possible, even in partially synchronous setting w.r.t. Δ (trivially EAAC)
1/3
ρ-bounded adversary, ρ in [1/3,2/3)
consistency violations possible and can be asymmetrically punished in the synchronous setting w.r.t. Δ* (non-trivially EAAC)
2/3
ρ-bounded adversary, ρ ≥ 2/3
consistency violations possible and cannot be asymmetrically punished, even in synchronous setting (not EAAC)
1

Fig. 4. Theorem 5.1. Non-trivial EAAC guarantees are possible in the quasi-permissionless and synchronous setting. The parameter Δ is an upper bound on message delays during "normal operation," while Δ^* bounds the time required for honest players to communicate (over the network or out-of-band) when the protocol is under attack.

Achieving asymmetric punishment via slashing in a proof-of-stake protocol would seem to require addressing the following challenges.

(1) There should be a "smoking gun" behind every consistency violation, in the form of a "certificate of guilt" that identifies (at least some of) the Byzantine players responsible for the violation.
(2) All honest players should learn such a certificate of guilt promptly after a consistency violation, before the adversary has had the opportunity to cash out its assets.
(3) Given the prompt receipt of a certificate of guilt, honest players should be able to reach consensus on a new (post-slashing) state.

Further, the second and third challenges must be met despite interference from an adversary that is so large as to be able to cause consistency violations.

At a very high level, our protocol resolves these three challenges as follows. (Section 6 provides a more technical overview of the protocol, with the full details deferred to the online version of the paper. The protocol is designed to be as simple as possible subject to a non-trivial EAAC guarantee; optimizing the performance of such protocols is an interesting direction for future work.) Some PBFT-style protocols provide certificates of guilt in the form of votes on conflicting blocks, and accordingly our starting point is the Tendermint protocol [Buchman, 2016, Buchman et al., 2018]. (Extending the permissioned Tendermint protocol to a proof-of-stake protocol with guaranteed consistency and liveness in the quasi-permissionless setting is non-trivial, but we show that it can be done.) The standard Tendermint protocol uses two stages of voting per view, and we show that this fails to guarantee the prompt receipt of certificates of guilt by honest players. On the other hand, we show that, provided the adversary controls less than two-thirds of the stake, three stages of voting suffice for the prompt dissemination of certificates of guilt. Finally, after a consistency violation, honest players attempt to reach consensus on an updated genesis block (in which slashing has been carried out) using a variant of the Dolev-Strong protocol [Dolev and Strong, 1983].

Interpretation for Ethereum. Our results make precise a number of common beliefs about the Ethereum protocol (post-merge). Most obviously, our work formalizes the potential security benefits of proof-of-stake sybil-resistance coupled with slashing logic.

Our impossibility result for the dynamically available setting (Theorem 4.1) shows that such security guarantees are impossible both for any protocol that relies on only off-chain resources (such as proof-of-work protocols) and for any standard longest-chain protocol (even if it is a proof-of-stake protocol). Thus, two of the biggest changes made to the Ethereum protocol during the merge—the switch from proof-of-work to proof-of-stake, and the addition of the Casper finality gadget [Buterin and Griffith, 2017]—are both necessary for provable slashing guarantees (neither change alone would enable asymmetric punishment). Our result also provides a justification for the "inactivity leaks" used in the Ethereum protocol to punish seemingly inactive players, which can be viewed as an economic mechanism for enforcing the (necessary) assumptions of the quasi-permissionless setting.

Our impossibility result for the partially synchronous setting (Theorem 4.2) justifies the common assumption that honest Ethereum validators could, in the case of emergency, communicate out of band within some known finite amount of time. Further, this result justifies the long "cooldown period" for unstaking in post-merge Ethereum, with a delay that is roughly proportional to the assumed time required for out-of-bound communication.

The protocol we design to prove our positive result (Theorem 5.1) resembles post-merge Ethereum in several high-level respects: the use of proof-of-stake sybil-resistance, an accountable PBFT-type approach to consensus (in our case, Tendermint rather than Casper), slashing for asymmetric punishment, equal-size validators, and regularly scheduled updates to the validator set. One notable difference is our protocol's reliance on three stages of voting to ensure the prompt dissemination of certificates of guilt. The reason for this is discussed in Section 9 of the online version of the paper, which can be found at https://arxiv.org/abs/2405.09173.

1.4 Discussion

The economic cost of controlling resources. This paper follows the tradition of Nakamoto [Nakamoto et al., 2008] (as in the quote in Section 1.2), among many others, in concentrating on the economic consequences to an attacker of causing a consistency violation (e.g., in order to execute a double-spend attack) in a permissionless consensus protocol. Our focus complements previous work of Budish [Budish, 2018, 2023], which provides an economic model for quantifying the cost that a potential attacker must absorb to control sufficient resources to be able to cause a consistency violation. The main conclusion in Budish [Budish, 2018, 2023] is that, for the current major blockchain protocols, this cost is surprisingly cheap, scaling as a flow cost rather than a stock cost. This analysis suggests that the economic consequences that *follow* an attack are the ones with the greatest potential to deter an attacker, and in this sense provides strong economic justification for the focus of this paper.

Consistency vs. liveness violations. One reason we focus on consistency violations (as opposed to liveness violations) is that they are the ones that can be unequivocally attributed to deviations from the intended protocol by an attacker (as opposed to network delays or other vagaries outside the control of honest participants). A second is that, in practice, consistency violations are generally considered much more serious than liveness violations (akin to money permanently disappearing from your bank account vs. the bank's computer system going down for a few hours).

2 The model

Impossibility results such as Theorems 4.1 and 4.2 require a precise model of the permissionless consensus protocol design space; we adopt the one defined by Lewis-Pye and Roughgarden [Lewis-Pye and Roughgarden, 2023]. In brief:

- There is a set of players (unknown to the protocol). Each player can control an unbounded number of identifiers and at each timeslot could be active or inactive.
- Messages are disseminated (e.g., via a gossip protocol) rather than sent point-to-point. Disseminated messages are eventually received by their recipients, possibly after a delay.
- In the synchronous setting, there is a finite upper bound, known to the protocol, on the worst-possible message delay. In the partially synchronous setting, there is an initial period of unknown finite duration in which message delays may be arbitrary.

- The behavior of a player is a function of the timeslot, its internal state, and the information it has received to-date from other players and from "oracles" (described below). Formally, a protocol is a specification of (the intended version of) this function.
- Each player maintains a running (ordered) list of transactions that it regards as confirmed.
- Players can own resources, which may be "external" to the protocol (e.g., hashrate) or "on-chain" (e.g., registered stake). External resources evolve independently of the protocol, while the values of on-chain resources generally depend on the transactions that have been confirmed thus far.
- A ρ-bounded adversary is one that never controls more than a ρ fraction of the overall amount of an (external or on-chain) resource that is controlled by the currently active players.
- Protocols may make use of "oracles" that represent cryptographic primitives. External resources are modeled as a special type of oracle (called a "permitter") to which the allowable queries depend on a player's resource balance.
- A protocol satisfies consistency if players' running transaction lists are consistent across players and across time (if T and T' are the lists of honest players p and p' at timeslots t and t', then either T is a prefix of T' or vice versa).
- A protocol satisfies liveness if, during periods of synchrony, honest players regularly add new confirmed transactions to their running lists.

Sections B.1–B.6 provide the mathematical details, and further discussion of the model can be found in [Lewis-Pye and Roughgarden, 2023].

In addition to this design space, we adopt the "hierarchy of permissionlessness" proposed in [Lewis-Pye and Roughgarden, 2023]. Intuitively, and phrased here specifically for proof-of-stake protocols, the two key definitions are the following (see Section B.7 for details):

- In the dynamically available setting, at every timeslot, at least one non-malicious player with a non-zero amount of registered stake is active.
- In the quasi-permissionless setting, at every timeslot, every non-malicious player with a non-zero amount of registered stake is active.

Appendix B fills in the details of these definitions.

3 The cost of an attack

3.1 An Overly Simplified Attempt

Consider an attacker poised to execute an attack, in the form of a consistency violation, on a permissionless consensus protocol. When would carrying out such an attack have negative economic consequences for the attacker, meaning that, ignoring off-chain gains (from double-spends, short positions, etc.), it's "worse off" than before? (Ideally, with honest players "no worse off" than before.) A first cut might be to track the market value of all the protocol-relevant resources owned by each (honest or Byzantine) player. That is, consider a blockchain protocol that uses k resources, and let $\mathsf{R}_i(p,t)$ denote the number of units of the ith resource (e.g., ASICs or coins) owned by player p at timeslot t. Let $C_i(t)$ denote the per-unit market price of the ith resource at timeslot t and define p's "net worth" at timeslot t by

$$V(p,t) := \sum_{i=1}^{k} \mathsf{R}_i(p,t) \cdot C_i(t). \tag{1}$$

The "consequences of an attack" carried out by a set B of Byzantine players at some timeslot t^* could then be measured by the value of B's resources immediately before and after the attack:

$$\frac{\sum_{p \in B} V(p, t^*_+)}{\sum_{p \in B} V(p, t^*_-)}, \tag{2}$$

with lower values of this ratio corresponding to more severe economic consequences of an attack. The idea that "the honest players H should be no worse off" would then translate to the condition that

$$V(p, t^*_+) \geq V(p, t^*_-) \tag{3}$$

for every $p \in H$.

We can then consider how the different scenarios laid out in Section 1.2 would translate to this formalism.

Scenario 1: status quo. Suppose the Bitcoin protocol suffers a consistency violation and yet neither of the narratives in Section 1.2 plays out as expected, with both the cryptocurrency price and the cryptographic hash function used for proof-of-work mining unchanged following the attack. Then the new market price $C(t^*_+)$ of an ASIC would equal the old price $C(t^*_-)$ and hence $V(p, t^*_+)$ would equal $V(p, t^*_-)$ for every player p. (In this example, there's only one resource—hashrate—and so we drop the dependence on i.) Thus, while the condition (3) would hold (which is good), the ratio in (2) would be 1, indicating an attack without any economic consequences. We will call such an attack *cheap*.

Scenario 2: price collapse (proof-of-work). Nakamoto's original narrative posited that a double-spend attack on the Bitcoin protocol would significantly decrease the USD value of BTC (and hence of ASICs for Bitcoin mining), an assumption that translates to $C(t^*_+) \ll C(t^*_-)$ and hence, for any (honest or Byzantine) ASIC-owning miner p, $V(p, t^*_+) \ll V(p, t^*_-)$. In this case, the attack is *expensive*, meaning that the ratio in (2) is less than one. It is expensive for the wrong reasons, however, in the sense that the conditon in (3) fails. We therefore say that the attack is *expensive due to collapse*.

Scenario 3: a hard fork (proof-of-work). The second narrative in Section 1.2, in which a double-spend attack on a proof-of-work protocol is punished through a hard fork that changes the cryptographic hash function used for proof-of-work mining, is mathematically equivalent to the first, with existing ASICs losing much of their value and so $C(t^*_+) \ll C(t^*_-)$. This again is an example of an attack that is expensive, but expensive due to collapse.

Scenario 4: price collapse (proof-of-stake). Consider now a proof-of-stake protocol—be it longest-chain, PBFT-type with slashing, or anything else—in which case $C(t)$ denotes the USD-denominated market price at time t of one coin of the protocol's native cryptocurrency. If a double-spend attack harms this cryptocurrency's price, then, as in Scenario 2, the attack is expensive due to collapse.

Scenario 5: successful slashing. Can any protocol—a proof-of-stake protocol, say—make attacks expensive for the right reasons, not due to collapse? To make this question precise, let's assume that a double-spend attack has no effect on the price of the protocol's native currency, i.e., $C(t^*_+) = C(t^*_-)$. Suppose further that a protocol is able to fulfill the promises of slashing outlined in Section 1.2, identifying (at least some of) the Byzantine players and surgically confiscating their stake (without inadvertently destroying any stake owned by honest players). Then, $\mathsf{R}(p, t^*_+)$ would be less than $\mathsf{R}(p, t^*_-)$ (and hence $V(p, t^*_+)$ would be less than $V(p, t^*_-)$) for at least some $p \in B$, while $V(p, t^*_+) = V(p, t^*_-)$ for all $p \in H$. As a result, the ratio in (2) would be less than 1 even as the condition in (3) holds. In this case, we call the attack *expensive even in the absence of collapse.* A protocol would then be called *EAAC* if every possible consistency violation would be expensive in this sense.

3.2 The Formal Definitions

Delayed economic consequences. The definitions proposed in Section 3.1 convey the spirit of our approach, but they are inadequate for a number of reasons. For example, there is no hope of designing a protocol that is EAAC in the sense above: the expression in (2) considers only the immediate economic cost suffered by an attack, while any protocol-inflicted punishment would require some time to take effect. For example, to enact slashing in a proof-of-stake protocol, honest players need time to communicate, confirm evidence of a consistency violation, and carry out the appropriate slashing instructions. (Changes in the price of the protocol's native currency, should they occur, would presumably also play out over a period of time.) Thus, we must instead insist that Byzantine players suffer economic consequences from a consistency violation at some timeslot t_f after the timeslot t^* in which the consistency violation is seen by honest players (here the "f" stands for "final"). Intuitively, t_f is a timeslot by which the aftermath of the attack has played out, with the currency price re-stabilized (possibly at a new level) and any protocol-inflicted punishment complete.

Investment functions. Next, the definition of the functions $\mathsf{R}_i(p, t)$, henceforth called *investment functions*, requires more care. For an external resource i, $\mathsf{R}_i(p, t)$ denotes the (objective) number of units of that resource owned by p at t, as before. For a resource i that corresponds to some form of stake or other on-chain resource, $\mathsf{R}_i(p, t)$ may depend on the current state of the blockchain protocol that controls the resource, meaning the sets of messages that have been received by each player by timeslot t. (We generally abuse notation and continue to write $\mathsf{R}_i(p, t)$, suppressing the dependence on the sets of received messages and the protocol that generates them.) We allow flexibility in exactly how investment functions are defined, but the rough idea is that $\mathsf{R}_i(p, t)$ should update once a resource-changing transaction has been "sufficiently processed" by the protocol (e.g., confirmed by an appropriate honest player). Our three main results concern investment functions of the following types:

- Our negative result for the dynamically available setting (Theorem 4.1) holds for any choice of investment functions, and thus requires no further modeling.
- Our negative result for the partially synchronous setting (Theorem 4.2) holds assuming only that, provided no consistency violation has been seen, zeroing out one's stake balance eventually zeroes out the corresponding investment function (i.e., the user eventually recoups their investment).

  Formally, consider an investment function R_i that corresponds to a stake allocation function S. Suppose that, in any execution E in which:
  - t is a timeslot $\geq$GST;
  - no honest player has seen a consistency violation by timeslot $t + \Gamma$;
  - for some player p and honest player p', $\mathsf{S}(\mathsf{S}^*, \mathsf{T}_{t'}, id) = 0$ for all $id \in \mathtt{id}(p)$ and $t' \in [t, t+\Gamma]$, where $\mathsf{T}_{t'}$ denotes the transactions confirmed for p' at timeslot t';

  it holds that $\mathsf{R}_i(p, t+\Gamma) = 0$ and, moreover, that $\mathsf{R}_i(p, t+\Gamma) = 0$ also in every execution that is indistinguishable from E for p' up to timeslot $t + \Gamma$ (i.e., every execution in which p' receives the same inputs, messages, and oracle responses at each timeslot $< t + \Gamma$ as in E). In this case, we call R_i a Γ-*liquid investment function.* (Intuitively, barring a consistency violation seen by honest players, p can eventually recoup its investment from p', or more generally from any third party that uses p' as its "source of truth" about which transactions have been confirmed.)

- For our positive result in Theorem 5.1, we use what we call a *canonical PoS* investment function. Here, there is a single resource, representing the amount of stake that a player has locked up in escrow. The resource balance of a player changes only through staking and unstaking transactions. A canonical investment function $\mathsf{R}(p, t)$ increases by x after a valid staking transaction (with staking amount x) is first confirmed by some honest player (possibly after a delay), and decreases by x after a valid unstaking transaction is first confirmed by some honest player (again, possibly after a delay). Such a function is Γ-liquid, where Γ is the maximum delay before the investment function reflects a newly confirmed staking or unstaking transaction.

In all these cases, we can interpret the economic investment of a player p at some timeslot t by the expression in (1), as before.

Valuation functions. Finally, we must specify the allowable behavior of a value function that represents the economic value of on-chain resources following a consistency violation. In general, we allow a quite abstract notion of a *valuation function* v that is a nonnegative function of a set P of players, a timeslot t, player investments at that time (the $\mathsf{R}_i(p, t)$'s), market prices at that time (the $C_i(t)$'s), and the sets of messages that have been received by each player by that time. Our impossibility results (Theorems 4.1 and 4.2) apply to all such valuations, and thus require no further modeling.

For our positive result (Theorem 5.1), we use what we call a *canonical PoS* valuation function, which imposes two much stronger restrictions (thereby strengthening the result). First, we require it to be *consistency-respecting*, in the sense that it is defined only for timeslots t at which consistency holds (i.e., if T and T' are the confirmed transactions for honest p and p' at t, then one of T, T' is a prefix of the other).[5] Second, at timeslots at which consistency does hold, the valuation function equals the value of the as-yet-unslashed stake-in-escrow of the players in question (at the current market prices).[6]

EAAC protocols. We now proceed to our formal definition of an "EAAC" protocol. As noted in the last scenario of Section 3.1, to have any hope of avoiding collateral damage to honest participants, we must assume that a consistency violation does not affect the market prices of the relevant resources. Thus, in the following definition, we consider a fixed price C_i for each resource rather than an arbitrary price sequence $\{C_i(t)\}_{t\geq 1}$.

Definition 3.1. A protocol is *EAAC with respect to investment functions* $\{\mathsf{R}_i(\cdot,\cdot)\}_{i\in[k]}$ *and valuation function* v in a given setting if, for every choice $\{C_i\}_{i\in[k]}$ of fixed resource prices, every external resource i, and every choice $\{\mathsf{R}_i(p, t)\}_{p\in P, t\geq 1}$ of players' investments in i:

(a) for every execution of the protocol consistent with the setting and every timeslot $t \geq 1$,

$$v(H, t, \{\mathsf{R}_i(p, t)\}_{p\in H, i\in[k]}, \{C_i\}_{i\in[k]}, \mathcal{M}) \geq \sum_{p\in H} \sum_{i=1}^{k} \mathsf{R}_i(p, t) \cdot C_i, \tag{4}$$

where H denotes the set of honest players and $\mathcal{M}$ the sets of messages received by each player by timeslot t in the execution; and

[5]Without this requirement, a protocol could achieve the EAAC property by addressing only the first two challenges from Section 1.3 and not the third. For example, the valuation function could be defined to discount any stake implicated by some certificate of guilt known to some honest player, even while honest players disagree on what the "post-slashing" state should be.

[6]See Section 9.9 of the online version of the paper for the precise definition used in the proof of Theorem 5.1.

(b) for every execution of the protocol consistent with the setting with a consistency violation that is first seen by honest players at a timeslot t^*, there exists a timeslot $t_f \geq t^*$ such that:

$$v(B, t_f, \{\mathsf{R}_i(p, t_f)\}_{p\in B, i\in[k]}, \{C_i\}_{i\in[k]}, \mathcal{M}) < \sum_{p\in B}\sum_{i=1}^{k} \mathsf{R}_i(p, t_f) \cdot C_i, \tag{5}$$

where B denotes the set of Byzantine players and $\mathcal{M}$ the sets of messages received by each player by timeslot t in the execution.[7]

Intuitively, part (a) of Definition 3.1 requires that honest players can always recoup whatever they may have invested (e.g., they are never slashed in a proof-of-stake protocol). Part (b) asserts that an attacker will be unable to fully cash out, with some of their investment lost to the protocol.

For brevity, we sometimes denote by α_H the ratio between the left- and right-hand sides of (4), and by α_B the analogous ratio for (5). (If the right-hand side is 0 or the left-hand side is undefined, we can interpret the ratio as 1.) Thus, the EAAC condition states that $\alpha_H \geq 1$ should always hold (i.e., if attacks are expensive, it's for the right reasons, not due to collapse) and that $\alpha_B < 1$ should hold at some point after a consistency violation (i.e., attacks are indeed expensive). If Definition 3.1 holds with condition (b) always satisfied with some $\alpha_B \leq \alpha < 1$, then we say that the protocol is *α-EAAC* (with respect to $\{\mathsf{R}_i(\cdot,\cdot)\}_{i\in[k]}$ and v).[8]

4 Impossibility results

Our first result establishes that blockchain protocols that are live and consistent in the dynamically available setting are always cheaply attackable in the absence of collapse. In other words, once an adversary is large enough to cause consistency violations, they are also large enough to avoid asymmetric punishment. The proof of Theorem 4.1 appears in Section C.

Theorem 4.1 (Impossibility Result for the Dynamically Available Setting). *In the dynamically available setting, with a $\frac{1}{2}$-bounded adversary, for every choice of investment functions and valuation function, no protocol can be live and EAAC. This holds even in the synchronous model and with Byzantine players that have fixed (i.e., time-invariant) resource balances.*

Recall the hierarchy of settings described in Section B.7. Theorem 4.1 establishes that, if we want to work with protocols that are non-trivially EAAC in some level of this hierarchy, then we cannot work in the fully permissionless or dynamically available settings and must instead focus on the quasi-permissionless or permissioned settings.

The next theorem shows that, even should we work in the quasi-permissionless setting, protocols cannot be non-trivially EAAC if we work in the partial synchrony model. The proof of Theorem 4.2 appears in Section 7.

Theorem 4.2 (Impossibility Result for the Partially Synchronous Setting). *In the quasi-permissionless setting and the partial synchrony model, with a $1 - 2\rho_l$-bounded adversary, for every choice of liquid investment functions and valuation function, no protocol can be ρ_l-resilient for liveness and EAAC.*

For example, suppose we restrict attention to the standard case of protocols which have the same resilience for liveness and consistency. The seminal result of Dwork, Lynch and Stockmeyer [Dwork et al., 1988] establishes that protocols for the partial synchrony model can be ρ-resilient only for

[7]For the purposes of inequalities (4) and (5), we interpret an undefined valuation function as $+\infty$.

[8]While "1" is arguably the most natural threshold when using these ratios to define "expensive" and "collapse," a different constant $x \in (0, 1)$ could be used instead. Our impossibility results would then hold, with essentially the same proofs, for any choice of x.

$\rho < 1/3$. Theorem 4.2 establishes that, if the adversary can own at least 1/3 of the resources, then the protocol cannot be EAAC. In fact, the proof establishes a stronger result: a $\frac{1}{3}$-bounded adversary can cause consistency violations without any punishment, asymmetric or otherwise (i.e., can carry out attacks that are cheap even when there is a collapse in the value of resources after the consistency violation is seen).

5 Provable slashing guarantees: a possibility result

Theorem 4.2 might seem to end the quest for PoS protocols with 'slashing' in the partial synchrony model. To circumvent this difficulty, we can consider protocols that are live and consistent in partial synchrony (for some small Δ) so long as the adversary is ρ-bounded for $\rho < 1/3$, and which, furthermore, implement slashing when the adversary owns more than 1/3 of the resources but message delivery prior to GST always occurs within some large known time bound Δ^* (*which may not be* $O(\Delta)$). We argue that this is a realistic setting: while blockchain protocols are commonly expected to be live and consistent in the partial synchrony model with a liveness parameter T_l determined by some small value of Δ (of the order of 1 second, say), it may also be reasonable to assume that messages will always eventually be delivered, either via the communication network or some "out-of-band" mechanism, within some sufficiently large time bound (of the order of a day or a week, say).[9] Theorem 4.2 does not rule out non-trivial EAAC protocols in this setting, provided that the time required for an attacker to recoup its investment following an attack scales with the worst-case delay Δ^*; see also the discussion at the end of Section 7.

Formally, we say a blockchain protocol is (ρ, ρ^*)*-EAAC* with respect to investment functions $\{\mathsf{R}_i(\cdot,\cdot)\}_{i\in[k]}$ and valuation function v if it satisfies the following two conditions, given any determined values of Δ and $\Delta^* \geq \Delta$, and where ρ^* is determined:

(i) The protocol is live and consistent for the partial synchrony model with respect to Δ, so long as the adversary is ρ-bounded. Here, we allow (as in Section B.5) the liveness parameter T_l to depend on Δ and other determined inputs, but require it to be independent of Δ^*. *In particular,* Δ^* *may be much larger than* T_l.

(ii) The protocol is EAAC (with respect to $\{\mathsf{R}_i(\cdot,\cdot)\}_{i\in[k]}$ and v) so long as the adversary is ρ^*-bounded and message delays prior to GST are always at most Δ^* (i.e., if p disseminates m at t, and if $p' \neq p$ is active at $t' \geq t + \Delta^*$, then p' receives that dissemination at a timeslot $\leq t'$).[10]

If in (ii) the protocol is α-EAAC in the sense of Section 3.2, then we say that the protocol is (α, ρ, ρ^*)-EAAC (with respect to $\{\mathsf{R}_i(\cdot,\cdot)\}_{i\in[k]}$ and v).

Our main positive result is the following. Canonical PoS investment and valuation functions are defined informally in Section 3.2, and formally in Section 9 of the online version of the paper.

Theorem 5.1 (Non-Trivial EAAC Protocols in the Quasi-Permissionless Setting). *For every* $\rho < 1/3$ *and* $\rho^* < 2/3$*, there exists a PoS protocol for the quasi-permissionless setting that is* (α, ρ, ρ^*)*-EAAC with respect to a canonical PoS investment function and a canonical PoS valuation, where*

$$\alpha = \max\{0, (\rho^* - \tfrac{1}{3})/\rho^*\}.$$

This result is optimal in several senses. First, an easy adaptation of the classic proof of Dwork, Lynch and Stockmeyer [Dwork et al., 1988] shows that protocols cannot be (ρ, ρ^*)-EAAC for

[9]We note also that our proof of Theorem 5.1 only requires that message delays prior to GST are bounded by Δ^* for a limited period around the time of an attack. This observation is made precise in Section 9 of the online version of the paper.

[10]The fact that Δ^* may be much larger than T_l means that a protocol aiming to be (ρ, ρ^*)-EAAC cannot ignore Δ and simply assume that message delays will always be bounded by Δ^*.

$\rho \geq 1/3$. Second, the bound of 2/3 on ρ^* is tight: an argument of Tas et al. [Tas et al., 2023], when translated to our framework, implies that, even in the synchronous setting, a protocol that is ρ_l-resilient for liveness cannot be EAAC with a $(1 - \rho_l)$-bounded adversary. The 2/3 bound in the statement of the theorem can therefore only be improved by weakening liveness requirements (we describe how our protocol can be so modified in Section 9 of the online version of the paper). Third, the bound on α cannot be improved: an adversary with a ρ^* fraction of the overall resources can cause a consistency violation using only one-third of the overall resources in a dishonest way [Dwork et al., 1988]; because honest players cannot be punished, the adversary can guarantee that it retains a $(\rho^* - \frac{1}{3})$ fraction of the overall original resources following its attack.

The proof of Theorem 5.1 appears in Section 9 of the full online version of the paper, which can be found at https://arxiv.org/abs/2405.09173. Section 6 provides a brief overview.

6 Overview of the PosT protocol and the proof of Theorem 5.1

We describe a PoS version of Tendermint, which we refer to as PosT. To specify PosT, we envisage that players can add and remove stake from a special staking contract, allowing them to act as 'validators'. While added to the staking contract, stake is regarded as being 'in escrow' and cannot be transferred between players. Removal of stake from escrow is subject to a delay (of time more than Δ^*). The stake allocation function S indicates the stake that an identifier has in escrow and which has not been marked for removal from escrow. This is the balance that describes an identifier's *weight* as a validator: we'll refer to an identifier's S-balance as their *validating stake*. Under the assumptions of the quasi-permissionless setting (see Section B.7), all honest players that possess a non-zero amount of validating stake at some timeslot are active at that timeslot.

The use of epochs. Recall that the instructions for Tendermint are divided into *views*. In each view, a designated *leader* proposes the next block of transactions, and other validators may vote on the proposal to form (one or two) *quorum certificates* (QCs) for the block. The instructions for PosT are divided into *epochs*, where each epoch is a sequence of consecutive views. Removal of stake from escrow is achieved via confirmation of an appropriate transaction. If this transaction is confirmed by the start of epoch e, then the corresponding stake remains in escrow until the end of the epoch, but no longer contributes to the weight of the corresponding identifier as a validator (their S-balance). It is only at the end of each epoch that the S-balance of an identifier can change, meaning that the set of validators is fixed during each epoch.

Certificates of guilt. Recall that $\rho^* < 2/3$ is determined and that our protocol is required to be EAAC only so long as the adversary is ρ^*-bounded. We use standard methods (e.g., see [Sheng et al., 2021]) to ensure that any consistency violation causes the production of a *certificate of guilt*, which is a set of signed messages proving Byzantine action on the part of some of the validating stake during a certain epoch. If a consistency violation occurs in some least epoch e, then the fact that $\rho^* < 2/3$ becomes crucial. This means that honest stake must participate in the confirmation of each block while in epoch e. If message delay prior to GST is always at most Δ^*, then honest validators will end epoch e within time Δ^* of each other and, by modifying Tendermint to utilize three stages of voting rather than two,[11] we will be able to argue that a certificate of guilt must be received by all active honest players within time $2\Delta^*$ of any honest player entering epoch $e + 1$. Defining epochs to be of sufficient length therefore ensures that any consistency violation in epoch e will produce a certificate of guilt which is seen by all active honest players before the end of epoch $e + 1$, and before the stake used to create the consistency violation is removed from escrow.

[11]HotStuff [Yin et al., 2019] also makes use of three stages of voting in each view but the motivation in that case is rather different.

The recovery procedure. The remaining challenge is to design a recovery procedure for honest players to use, after receiving a certificate of guilt, to reach consensus on a new state in which slashing has been carried out. In more detail, suppose a consistency violation occurs in some least epoch e, with all active honest players receiving a certificate of guilt before the end of epoch $e + 1$. The goal of the recovery procedure is for all honest validators (for epoch e) to output some common block b—an updated genesis block, in effect—which *includes all transactions confirmed by the end of epoch* $e - 1$, and which 'slashes' (renders unspendable) at least 1/3 of the validating stake for epoch e (with certificates of guilt provided for all slashed stake).

One potential difficulty in implementing such a recovery procedure is that a consistency violation is only guaranteed to produce a certificate of guilt for 1/3 of the validating stake. If the adversary possesses $\geq 5/9$ of the validating stake, then they may possess at least a fraction $(5/9-1/3)/(2/3) = 1/3$ of the validating stake that remains after slashing. If one were to naively employ some Tendermint-like protocol to reach consensus on an updated genesis block (including slashing conditions) for the next iteration of the protocol, such an adversary could threaten liveness.

The simplest approach to address this issue, and the one we follow here given our focus on fundamental possibility and impossibility results (as opposed to more fine-grained performance considerations), is to base the recovery procedure instead on the classic protocol of Dolev and Strong [Dolev and Strong, 1983], with delay Δ^* between each round of the protocol. In a first instance of the Dolev-Strong protocol, a designated *leader* is asked to propose an updated genesis block. If this instance results in consensus amongst honest players on such a block b, then the honest majority of validating stake that remains after slashing (since $\rho^* < 2/3$) can "sign off" on this value, producing a form of *certificate* that acts as proof that b can be used as an updated genesis block. If not, a second leader for another instance of the Dolev-Strong protocol is then asked to propose an updated genesis block, and so on. Whenever an honest leader is chosen (if not before), the corresponding instance of the Dolev-Strong protocol will conclude with consensus on an updated genesis block, from which the main protocol can then resume.[12]

The full details of the PosT protocol and the proof of Theorem 5.1 are provided in Section 9 of the online version of the paper.

7 Impossibility result for the partially synchronous setting: The proof of Theorem 4.2

Consider the quasi-permissionless setting and the partial synchrony model and suppose the blockchain protocol $(\Sigma, \mathcal{O}, \mathcal{C}, \mathcal{S})$ is ρ_l-resilient for liveness. We consider three non-empty and pairwise disjoint sets of players, X, Y and Z, such that:

- $\mathcal{P} = X \cup Y \cup Z$, $|\mathcal{P}| = n$ (say);
- $|X| = |Z| = \lfloor n\rho_l \rfloor$ and;
- $|Y| = n - 2\lfloor n\rho_l \rfloor$.

The intuition. For the sake of simplicity, consider a pure proof-of-stake protocol, meaning a protocol that does not make use of external resources (although the formal proof below treats the general case). Suppose all players begin with a single unit of each form of stake and are active at all timeslots. The players in X and Z are honest, while the players in Y are Byzantine.

Because we are in the partial synchrony model, we may suppose that messages disseminated by players in X are received by players in $X \cup Y$ at the next timeslot, but are not received by players in Z until after GST. Symmetrically, we may suppose that messages disseminated by players in Z

[12]For simplicity, our protocol description and analysis conclude with the successful agreement by honest players on a post-slashing state following a consistency violation. This suffices to establish the EAAC property and prove Theorem 5.1. More generally, the protocol could be run repeatedly, triggering the recovery procedure as needed to punish a consistency violation and produce a new genesis blocks for the next instance of the protocol.

are received by players in $Z \cup Y$ at the next timeslot, but are not received by players in X until after GST. Suppose that the players in Y initially act towards those in X as if GST$= 0$ but the players in Z are Byzantine and are not disseminating messages. Because the players in Z own at most a ρ_l fraction of the stake and the protocol is ρ_l-resilient for liveness, the players in X must eventually confirm a sequence of transactions, which may include transactions transferring all stake from players in Y. If the players in Y simultaneously act towards those in Z as if GST$= 0$ but the players in X are Byzantine and are not disseminating messages, then the players in Z must eventually confirm a sequence of transactions, which may include transactions transferring all stake from players in Y. When GST arrives, this means that the honest players see a consistency violation, but by this time the players in Y have already cashed out all of their stake.

The formal proof. Fix arbitrary Γ-liquid investment functions and an arbitrary valuation function. We consider three protocol executions in which all parameters remain the same unless explicitly stated otherwise. The player set is always $X \cup Y \cup Z$ (as described above), and each player uses a single identifier (which we identify with the player). All players begin with a single unit of each form of stake (if there exist any such) and are always active. Let $\mathtt{tr}_1, \mathtt{tr}_2$ denote distinct transactions that are benign in the sense of Section B.4 and preserve the total amount of resources controlled by the players in each of X, Y, and Z (e.g., a simple transfer between two players of X). For $i = 1, 2$, let T_i denote a set of transactions such that, no matter how they ordered, $\mathsf{S}_h(\mathsf{S}^*_h, \mathtt{tr}_i * \mathsf{T}_i, y) = 0$ for all stake allocation functions $\mathsf{S}_h \in \mathcal{S}$ and players $y \in Y$. (The sets T_1 and T_2 exist according to the assumptions in Section B.4.) For the sake of simplicity, we suppose that $\Delta = 1$, but the proof is easily adapted to deal with larger values for Δ.

Execution 1. GST$= 0$. Players in X and Y are honest. Players in Z are Byzantine and do not disseminate messages. Players in X have a single unit of each form of external resource at each timeslot, while players in Y and Z do not own external resources. The environment sends a single transaction $\mathtt{tr}_1$ to a player $p_1 \in X$ at timeslot 1. At timeslot $2T_l$, the environment sends the transactions in T_1 to p_1.

Execution 2. This is similar to Execution 1, but with the roles of X and Z reversed. GST$= 0$. Players in Y and Z and honest. Players in X are Byzantine and do not disseminate messages. Players in Z have a single unit of each form of external resource at each timeslot, while players in X and Y do not own external resources. The environment sends a single transaction $\mathtt{tr}_2$ to a player $p_2 \in Z$ at timeslot 1. At timeslot $2T_l$, the environment sends the transactions in T_2 to p_2.

Execution 3. The execution is specified as follows:

- Players in $X \cup Z$ are honest, while players in Y are Byzantine.
- GST$=3T_l + \Gamma$. Any message disseminated by a player in X at any timeslot t is received by players in $X \cup Y$ at the next timeslot, but is not received by players in Z until $\max\{\text{GST}, t+1\}$. Any message disseminated by a player in Z at any timeslot t is received by players in $Y \cup Z$ at the next timeslot, but is not received by players in X until $\max\{\text{GST}, t + 1\}$.
- Players in $X \cup Z$ have a single unit of each form of external resource at each timeslot, while players in Y do not own external resources.
- The environment sends $\mathtt{tr}_1$ to p_1 and $\mathtt{tr}_2$ to p_2 at timeslot 1. At timeslot $2T_l$, the environment sends the transactions in T_1 to p_1 and the transactions in T_2 to p_2. The environment then sends no further transactions.
- Each Byzantine player simulates two honest players. The first of these players acts honestly, except that they pretend messages sent by players in Z prior to GST do not arrive until GST. A message disseminated by this player at any timeslot t is received by players in $X \cup Y$ at the next timeslot, but is not received by players in Z until $\max\{\text{GST}, t + 1\}$. The second of these

simulated players acts honestly, except that they pretend messages sent by players in X prior to GST do not arrive until GST. A message disseminated by this player at any timeslot t is received by players in $Y \cup Z$ at the next timeslot, but is not received by players in X until $\max\{\text{GST}, t+1\}$.

Analysis. Note that, in Execution 1, the adversary is ρ_l-bounded. Since GST$=0$, $\mathtt{tr}_1$ must be confirmed for all honest players by timeslot $1+T_l$ (while $\mathtt{tr}_2$ is not, because this transaction is not received by any player). Similarly, the transactions sent to p_1 at $2T_l$ must be confirmed by $3T_l$; by the definition of T_1, players in Y own no stake (of any form) at timeslots $\geq 3T_l$. Because the investment functions corresponding to $\mathsf{S}_1, \dots, \mathsf{S}_j$ are assumed to be Γ-liquid, $\mathsf{R}_h(p, 3T_l + \Gamma) = 0$ for all such investment functions and all $p \in Y$.

In Execution 2, the adversary is also ρ_l-bounded. Since GST$=0$, $\mathtt{tr}_2$ must be confirmed for all honest players by timeslot $1+T_l$ (while $\mathtt{tr}_1$ is not, because this transaction is not received by any player). Similarly, the transactions sent to p_2 at $2T_l$ must be confirmed by $3T_l$, meaning that players in Y own no stake (of any form) at timeslots $\geq 3T_l$. Because the investment functions corresponding to $\mathsf{S}_1, \dots, \mathsf{S}_j$ are assumed to be Γ-liquid, $\mathsf{R}_h(p, 3T_l + \Gamma) = 0$ for all such investment functions and all $p \in Y$.

To complete the argument, note that, for players in X, Execution 3 is indistinguishable from Execution 1 at all timeslots <GST, i.e. those players receive precisely the same inputs, messages and oracle responses at all timeslots <GST. It follows that $\mathtt{tr}_1$ is confirmed for all players in X by timeslot $1+T_l$, and that the transactions of T_1 are confirmed for all players in X by timeslot $3T_l$. Similarly, for players in Z, Execution 3 is indistinguishable from Execution 2 at all timeslots <GST. It follows that $\mathtt{tr}_2$ is confirmed for all players in Z by timeslot $1+T_l$, and that the transactions of T_2 are confirmed for all players in Z by timeslot $3T_l$. It further follows that, in Execution 3, $\mathsf{R}_h(p, 3T_l + \Gamma) = 0$ for every investment function R_h and $p \in Y$.[13]

A consistency violation is first seen by honest players at GST. For all choices of t_f ≥GST, the Byzantine players have already cashed out their stakes (and never possessed any external resources):

$$\sum_{p \in Y} \sum_{i=1}^{k} \mathsf{R}_i(p, t_f) \cdot C_i = 0$$

and hence $\alpha_B = 1$. The protocol therefore fails to be EAAC.

Interpretation for the synchronous model. As alluded to in Section 5, the proof of Theorem 4.2 continues to hold in the synchronous model if $3T_l + \Gamma$ is less than the worst-case message delay Δ. That is, to avoid the impossibility result in Theorem 4.2, either the time to transaction confirmation or the time to recoup an investment off-chain (following a transaction confirmation on-chain) must scale with the worst-case message delay. For example, if typical network delays are much smaller than worst-case delays and the speed of transaction confirmation in some PoS protocol scales with the former, then the "cooldown period" required before stake can be liquidated must scale with the latter (as it does, roughly, in the current Ethereum protocol).

Acknowledgments

The research of the third author at Columbia University is supported in part by NSF awards CCF-2006737 and CNS-2212745, and research awards from the Briger Family Digital Finance Lab and the Center for Digital Finance and Technologies.

[13]The definition of a Γ-liquid investment function allows a player to cash out after *some* honest player sees a zero balance on-chain for at least Γ consecutive time steps; here, in fact, *all* honest players witness this.

References

Joseph Bonneau. 2016. Why buy when you can rent? Bribery attacks on bitcoin-style consensus. In *International Conference on Financial Cryptography and Data Security*. Springer, 19–26.

Ethan Buchman. 2016. *Tendermint: Byzantine fault tolerance in the age of blockchains.* Ph. D. Dissertation.

Ethan Buchman, Jae Kwon, and Zarko Milosevic. 2018. The latest gossip on BFT consensus. *arXiv preprint arXiv:1807.04938* (2018).

Eric Budish. 2018. *The economic limits of bitcoin and the blockchain.* Technical Report. National Bureau of Economic Research.

Eric Budish. 2023. *Trust at scale: The economic limits of cryptocurrencies and blockchains.* Technical Report. Working Paper, U. of Chicago.

Vitalik Buterin. 2014a. Proof-of-stake: How I learned to love weak subjectivity. *https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity* (2014).

Vitalik Buterin. 2014b. Slasher: A Punitive Proof-of-Stake Algorithm. *https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm* (2014).

Vitalik Buterin. 2016. A proof of stake design philosophy. *https://medium.com/@VitalikButerin/a-proof-of-stake-design-philosophy-506585978d51* (2016).

Vitalik Buterin. 2017. Minimal slashing conditions. *https://medium.com/@VitalikButerin/minimal-slashing-conditions-20f0b500fc6c* (2017).

Vitalik Buterin and Virgil Griffith. 2017. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437* (2017).

Christian Cachin, Rachid Guerraoui, and Luís Rodrigues. 2011. *Introduction to reliable and secure distributed programming.* Springer Science & Business Media.

Miguel Castro, Barbara Liskov, et al. 1999. Practical byzantine fault tolerance. In *OsDI*, Vol. 99. 173–186.

Jing Chen, Sergey Gorbunov, Silvio Micali, and Georgios Vlachos. 2018. ALGORAND AGREEMENT: Super Fast and Partition Resilient Byzantine Agreement. *IACR Cryptol. ePrint Arch.* 2018 (2018), 377.

Jing Chen and Silvio Micali. 2016. Algorand. *arXiv preprint arXiv:1607.01341* (2016).

Pierre Civit, Seth Gilbert, and Vincent Gramoli. 2021. Polygraph: Accountable byzantine agreement. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 403–413.

Pierre Civit, Seth Gilbert, Vincent Gramoli, Rachid Guerraoui, and Jovan Komatovic. 2023. As easy as ABC: Optimal (A) ccountable (B) yzantine (C) onsensus is easy! *J. Parallel and Distrib. Comput.* 181 (2023), 104743.

Pierre Civit, Seth Gilbert, Vincent Gramoli, Rachid Guerraoui, Jovan Komatovic, Zarko Milosevic, and Adi Seredinschi. 2022. Crime and punishment in distributed byzantine decision tasks. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 34–44.

Bram Cohen and Krzysztof Pietrzak. 2019. The chia network blockchain. *vol* 1 (2019), 1–44.

Phil Daian, Rafael Pass, and Elaine Shi. 2019. Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake. In *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*. Springer, 23–41.

Soubhik Deb, Robert Raynor, and Sreeram Kannan. 2024. STAKESURE: Proof of Stake Mechanisms with Strong Cryptoeconomic Safety. *arXiv preprint arXiv:2401.05797* (2024).

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

Cynthia Dwork, Nancy A. Lynch, and Larry Stockmeyer. 1988. Consensus in the Presence of Partial Synchrony. *J. ACM* 35, 2 (1988), 288–323.

Joshua S Gans and Hanna Halaburda. 2023. *" Zero Cost"Majority Attacks on Permissionless Blockchains.* Technical Report. National Bureau of Economic Research.

Juan A Garay, Aggelos Kiayias, and Nikos Leonardos. 2018. The Bitcoin Backbone Protocol: Analysis and Applications. (2018).

Hanna Halaburda, Guillaume Haeringer, Joshua Gans, and Neil Gandal. 2022. The microeconomics of cryptocurrencies. *Journal of Economic Literature* 60, 3 (2022), 971–1013.

Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. 2017. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*. Springer, 357–388.

Jae Kwon. 2014. Tendermint: Consensus without Mining. (2014).

Jacob Leshno, Rafael Pass, and Elaine Shi. 2023. Can open decentralized ledgers be economically secure? *Cryptology ePrint Archive* (2023).

Andrew Lewis-Pye and Tim Roughgarden. 2023. Permissionless Consensus. *arXiv preprint arXiv:2304.14701* (2023).

Nancy A Lynch. 1996. *Distributed algorithms.* Elsevier.

Satoshi Nakamoto et al. 2008. Bitcoin: A peer-to-peer electronic cash system.(2008).

Joachim Neu, Ertem Nusret Tas, and David Tse. 2022. The availability-accountability dilemma and its resolution via accountability gadgets. In *International Conference on Financial Cryptography and Data Security*. Springer, 541–559.

Alejandro Ranchal-Pedrosa and Vincent Gramoli. 2020. ZLB: A blockchain to tolerate colluding majorities. *arXiv preprint arXiv:2007.10541* (2020).

Alex Shamis, Peter Pietzuch, Burcu Canakci, Miguel Castro, Cédric Fournet, Edward Ashton, Amaury Chamayou, Sylvan Clebsch, Antoine Delignat-Lavaud, Matthew Kerner, et al. 2022. {IA-CCF}: Individual Accountability for Permissioned Ledgers. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. 467–491.

Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. 2021. BFT protocol forensics. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*. 1722–1743.

Srivatsan Sridhar, Dionysis Zindros, and David Tse. 2023. Better Safe than Sorry: Recovering after Adversarial Majority. *arXiv preprint arXiv:2310.06338* (2023).

Ertem Nusret Tas, David Tse, Fangyu Gai, Sreeram Kannan, Mohammad Ali Maddah-Ali, and Fisher Yu. 2023. Bitcoin-enhanced proof-of-stake security: Possibilities and impossibilities. In *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 126–145.

Ertem Nusret Tas, David Tse, Fisher Yu, and Sreeram Kannan. 2022. Babylon: Reusing bitcoin mining to enhance proof-of-stake security. *arXiv preprint arXiv:2201.07946* (2022).

Maofan Yin, Dahlia Malkhi, Michael K Reiter, Guy Golan Gueta, and Ittai Abraham. 2019. HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*. 347–356.

A Further related work

A.1 Accountability: positive results

The closest analog to our work in the distributed computing literature is a sequence of papers, including Buterin and Griffith [Buterin and Griffith, 2017], Civit et al. [Civit et al., 2021], and Shamis et al. [Shamis et al., 2022], about protocols that satisfy *accountability*, meaning protocols that can provide certificates of guilt in the event of a consistency violation. Further examples of papers in this sequence include Sheng et al. [Sheng et al., 2021], who analyze accountability for well-known permissioned protocols such as HotStuff [Yin et al., 2019], PBFT [Castro et al., 1999], Tendermint [Buchman, 2016, Buchman et al., 2018], and Algorand [Chen et al., 2018]; and Civit et al. [Civit et al., 2023, 2022], who describe generic transformations that take any permissioned protocol designed for the partially synchronous setting and provide a corresponding accountable version.

None of these papers describe how to carry out asymmetric punishment (e.g., slashing) and thus fall short of our goals here—that is, they address challenge (1) from Section 1.3, but not challenges (2) and (3). The one exception to this point is the ZLB protocol of Ranchal-Pedrosa and Gramoli [Ranchal-Pedrosa and Gramoli, 2020], which is a permissioned blockchain protocol (with a fixed and known player set) that is able to implement slashing when the adversary controls less than a 5/9 fraction of the player set. Sridhar et al. [Sridhar et al., 2023] specify a "gadget" that can be applied to blockchain protocols operating in the synchronous setting to reboot and maintain consistency after an attack, but they do not implement slashing and assume that an honest majority is somehow reestablished out-of-protocol.

Even more significantly, with the exception of [Sridhar et al., 2023] and [Neu et al., 2022, Tas et al., 2023, 2022] (described below), the entire literature on accountability considers only permissioned protocols. Turning a permissioned protocol into a permissionless one is generally technically challenging (if not impossible) due to the three additional challenges listed in Section 1.1. For example, while Dwork et al. [Dwork et al., 1988] showed in 1988 how to achieve consistency and liveness in the partially synchronous setting with an adversary that controls less than one-third of the players, the first permissionless analog of this result was proved only last year [Lewis-Pye and Roughgarden, 2023]. Our positive result here (Theorem 5.1) provides a (permissionless) proof-of-stake protocol that, for an adversary controlling less than two-thirds of the total stake, provably implements slashing: It reaches consensus on slashing conditions, even in the face of consistency violations, before the adversary is able to remove their stake, thereby guaranteeing the EAAC property. As the protocol overview in Section 6 makes clear, a number of new ideas are required to obtain this result. Our positive result constitutes a significant advance even in the permissioned setting (relative to [Ranchal-Pedrosa and Gramoli, 2020]) in that our protocol can punish adversaries that control less than two-thirds (as opposed to five-ninths) of the overall player participation. As noted in Section 1.3, this bound of 2/3 is the best possible.

A.2 Accountability: negative results

The literature on accountability has focused primarily on positive results. One exception is Neu et al. [Neu et al., 2022], who prove that no protocol operating in the dynamically available setting can provide accountability. The authors then provide an approach to addressing this limitation by describing a "gadget" that checkpoints a longest-chain protocol. The "full ledger" is then live in the dynamically available setting, while the checkpointed prefix ledger provides accountability. Another exception is Tas et el. [Tas et al., 2023, 2022] who, in addition to positive results on defending against long-range attacks, prove negative results on accountability for adversaries that control at least two-thirds of the player set. None of these papers provide any economic modeling, which is a key contribution of the present work (see Section 3).

A.3 The cost of double-spend attacks

On the economics side, the most closely related work to ours is that of Budish [Budish, 2018, 2023], which is reviewed briefly in Section 1.4; see the references therein and Halaburda et al. [Halaburda et al., 2022] for a broader view of economics research on blockchain protocols. Bonneau [Bonneau, 2016], Leshno et al. [Leshno et al., 2023], and Gans and Halaburda [Gans and Halaburda, 2023] provide additional arguments that, at least for Bitcoin-like protocols, double-spend attacks may be very cheap to execute if there are no post-attack consequences. Leshno et al. [Leshno et al., 2023] also propose a variant of the Bitcoin protocol that halts whenever a consistency violation is detected (in effect, making consistency violations impossible by converting them into liveness violations).

A.4 The game theory of slashing

Deb et al. [Deb et al., 2024] consider the game theory of slashing from an angle that is complementary to ours, in a model in which the adversary need not own resources to carry out an attack, but can instead offer bribes to players. (Bonneau [Bonneau, 2016] and Leshno et al. [Leshno et al., 2023] similarly consider the possibility of bribery by an attacker.) The authors argue that, without slashing, rational players can be incentivized to accept small bribes to deviate from the protocol, even when such deviations by a large number of players causes a consistency violation and a collapse in the value of their resources. Fundamentally, the reason for this is that the price collapse is non-targeted, while bribe pay-offs depend directly on individual actions. The authors point out that targeted slashing, assuming it could be somehow implemented, could ensure that a unilateral deviation by a single player would lead to significant punishment by the protocol, in which case bribes would be effective only if they were large.

B Full Description of Model

B.1 The set of players and the means of communication

The set of players. We consider a potentially infinite set of players $\mathcal{P}$. Each player $p \in \mathcal{P}$ is allocated a non-empty and potentially infinite set of *identifiers* $\mathtt{id}(p)$. One can think of $\mathtt{id}(p)$ as an arbitrarily large pre-generated set of public keys for which p knows the corresponding private key; a player p can use its identifiers to create an arbitrarily large number of sybils. Identifier sets are disjoint, meaning $\mathtt{id}(p) \cap \mathtt{id}(p') = \emptyset$ when $p \neq p'$; intuitively, no player knows the private keys that are held by other players.

Permissionless entry and exit. Time is divided into discrete timeslots $t = 1, 2, \dots$, and each player may or may not be *active* at each timeslot. A *player allocation* is a function specifying $\mathtt{id}(p)$ for each $p \in \mathcal{P}$ and the timeslots at which each player is active. Because protocols generally require *some* active players to achieve any non-trivial functionality, we assume that a non-zero but finite number of players is active at each timeslot. The player allocation is exogenous, in that we do not model why a player might be active or inactive at a given timeslot.

Inputs. Each player is given a finite set of *inputs*, which capture its knowledge at the beginning of the execution of a protocol. If a variable is specified as one of p's inputs, we refer to it as *determined for p*, otherwise it is *undetermined for p*. If a variable is determined/undetermined for all p then we refer to it as *determined/undetermined.* For example, to model a permissionless environment with sybils, we assume that $\mathtt{id}(p)$ and the timeslots at which p is active are determined for p but undetermined for $p' \neq p$.

Message sending. At each timeslot, each active player may *disseminate* a finite set of messages (each of finite length), and will receive a (possibly empty) multiset of messages that have been disseminated by other players at previous timeslots. Inactive players do not disseminate or receive any messages. We consider two of the most common models of communication reliability, the synchronous and partially synchronous models. These models have to be adapted, however, to deal with the fact that players may not be active at all timeslots:
Synchronous model. There exists some determined $\Delta \in \mathbb{N}_{>0}$ such that if p disseminates m at t, and if $p' \neq p$ is active at $t' \geq t + \Delta$, then p' receives that dissemination at a timeslot $\leq t'$.
Partially synchronous model. There exists some determined $\Delta \in \mathbb{N}_{>0}$, and undetermined timeslot GST such that, if p disseminates m at t and $p' \neq p$ is active at $t' \geq \max\{\text{GST}, t\} + \Delta$, then p' receives that dissemination at a timeslot $\leq t'$.

B.2 Players and oracles

Using an approach that is common in distributed computing, we model player behavior via state machine diagrams. The description of a protocol also specifies a (possibly empty) set of *oracles* $O = \{O_1, \dots, O_z\}$, which are used to capture idealized cryptographic primitives. Players may send *queries* to the oracles and will then receive *responses* in return. At each timeslot t, the state transition made by player p therefore depends on the oracle responses received by p at t, as well as t, p's present state, and the messages received by p at t. We refer the reader to [Lewis-Pye and Roughgarden, 2023] for a simple description of how oracles can be used to model standard cryptographic primitives such as signature schemes, verifiable delay functions, and ephemeral keys. Section B.3 describes precisely how oracles can be used to model *external resources* such as ASICs (for proof-of-work protocols) and memory chips (for proof-of-space protocols).

Byzantine and honest players. To ensure that our impossibility results are as strong as possible, we consider a *static adversary.* In the static adversary model, each player is either *Byzantine* or *honest* and an arbitrary and undetermined subset of the players may be Byzantine. The difference between Byzantine and honest players is that honest players must have the state diagram specified by the protocol, while Byzantine players may have arbitrary state diagrams.[14] To model a perfectly

[14]While one might suppose that honest players are incentivized by inflationary rewards (e.g., block rewards or staking rewards paid out in newly minted coins by the protocol to those that appear to run it honestly) and Byzantine players are motivated by off-chain gains (e.g., profit from a double-spend attack or a judiciously chosen short position), we do not attempt to microfound why a given player might choose to be honest or Byzantine.

coordinated adversary, we also allow that the instructions carried out by each Byzantine player can depend on the messages and responses received by other Byzantine players. That is, if p is Byzantine, the messages p disseminates, the queries p sends, and p's state transition at a timeslot t are a function not only of p's state and the multiset of messages and oracle responses received by p at t, but also of the corresponding values for all the other Byzantine players.

B.3 Modeling external resources

External vs. on-chain resources. We suppose blockchain validation requires *resources*, which can either be *external* or *on-chain*. External resources are the hardware (such as ASICs or memory chips) required by validators in proof-of-work (PoW) or proof-of-space (PoSp) protocols. By contrast, on-chain resources are those such as stake that are recorded on the blockchain. From the perspective of our analysis here, a key distinction between external and on-chain resources is that the latter can be confiscated by consensus amongst validators.

External resources and permitters. *Permitter oracles*, or simply *permitters*, are required for modeling external resouces (but not for on-chain resources). A protocol may use a finite set of permitters. These are listed as part of the protocol amongst the oracles in $\mathcal{O}$, but have some distinguished features.

The resource allocation. For each execution of the protocol, and for each permitter oracle O, we are given a corresponding *resource allocation*, denoted R^O. We assume that R^O is undetermined, as befits an "external" resource. The resource allocation can be thought of as assigning each player some amount of the external resource at each timeslot. That is, for all $p \in \mathcal{P}$ and t, we have $\mathsf{R}^O(p,t) \in \mathbb{N}$. We refer to $\mathsf{R}^O(p,t)$ as p's *balance* at t. Because balances are unknown to a protocol, an inactive player might as well have a zero balance: $\mathsf{R}^O(p,t) = 0$ whenever p is not active at t. For each t, we also define $\mathsf{R}^O(t) := \sum_p \mathsf{R}^O(p,t)$.

Restricting the adversary. An arbitrary value $\mathsf{R}^O_{\max}$ is given as a protocol input. This value is determined, but the protocol must function for any given value of $\mathsf{R}^O_{\max} \geq 1$.[15] Let B denote the set of Byzantine players and define $\mathsf{R}^O_B(t) := \sum_{p \in B} \mathsf{R}^O(p,t)$. For $\rho \in [0,1]$, we say that R^O is *ρ-bounded* if the following conditions are satisfied for all t:

- $\mathsf{R}^O(t) \in [1, \mathsf{R}^O_{\max}]$.
- $\mathsf{R}^O_B(t)/\mathsf{R}^O(t) \leq \rho$.

The smaller the value of ρ, the more severe the restriction on the combined "power" of the Byzantine players. If all resource allocations corresponding to permitters in $\mathcal{O}$ are ρ-bounded, then we say the adversary is *externally ρ-bounded.*

Permitter oracles. At each timeslot, a player may send *queries* to each permitter oracle. These queries can be thought of as requests for proof-of-work or proof-of-space, and the player will then receive a *response* to each query at the same timeslot. The difference between permitter oracles and other oracles is that the queries that a player p can send to a permitter oracle O at timeslot t depend on $\mathsf{R}^O(p,t)$. If a player p sends a query to the permitter oracle O at timeslot t, the query must be of the form (b, σ) such that $b \in \mathbb{N}$ and $b \leq \mathsf{R}^O(p,t)$. Importantly, this constraint applies also to Byzantine players. In the case of a proof-of-work protocol, the query (b,σ) can be thought of as a request for a proof-of-work for the string σ, to which hashrate b is committed at timeslot t.

[15]The upper bound $\mathsf{R}^O_{\max}$ on the total resource balance can be very loose, for example representing all of the silicon on planet earth.

Single and multi-use permitters. A player may make multiple queries $(b_1, \sigma_1), \dots, (b_k, \sigma_k)$ to a permitter in a single timeslot t, subject to the following constraints. With a *single-use* permitter—the appropriate version for modeling the Bitcoin protocol—these queries are restricted to satisfy $\sum_{i=1}^{k} b_i \leq \mathsf{R}^O(p, t)$. (Interpreting the b_i's as hashrates devoted to the queries, this constraint asserts that none of a player's overall hashrate can be "reused.") With a *multi-use* permitter—the more convenient version for modeling protocols that incorporate proof-of-space such as Chia [Cohen and Pietrzak, 2019]—the only restriction is that $b_i \leq \mathsf{R}^O(p, t)$ for each $i = 1, 2, \dots, k$. (Intuitively, space devoted to storing lookup tables can be reused across different "challenges.")

Permitter responses. If O is deterministic, then when p sends the query (b, σ) to O at timeslot t, the values t, b, and σ determine the response r of the permitter. To prevent responses from being forged by Byzantine players, r is a message signed by the permitter. If O is probabilistic, then t, b, and σ determine a distribution on responses.

B.4 Modeling stake

For the sake of simplicity, we restrict attention here to on-chain resources that are forms of stake. We refer the reader to [Lewis-Pye and Roughgarden, 2023] for further discussion of other forms of on-chain resources, and note that the results we present here are easily adapted to accommodate general on-chain resources.

The approach to modeling stake. This section defines *stake allocation functions*, which take *transactions* as inputs. Section B.5 describes how *blockchain protocols* are required to select specific sets of transactions to which stake allocation functions can be applied so as to specify the stake of each player at a given point in a protocol execution.

The environment. For each execution of a blockchain protocol, there exists an undetermined *environment*, denoted En, which sends *transactions* to players.[16] Transactions are messages signed by the environment. If En sends $\mathtt{tr}$ to p at timeslot t, then p *receives* $\mathtt{tr}$ at t as a member of its multiset of messages received at that timeslot. Formally, the environment En is simply a set of triples of the form $(p, \mathtt{tr}, t)$ such that p is active at t. We stipulate that, if $(p, \mathtt{tr}, t) \in \mathsf{En}$, then p receives the transaction $\mathtt{tr}$ at t, in addition to the other disseminations that it receives at t. We assume that, for each $p \in \mathcal{P}$ and each t, there exist at most finitely many triples $(p, \mathtt{tr}, t)$ in En.

We allow a protocol to specify a finite set of *stake allocation functions*, representing one or more forms of stake (e.g., stake amounts held in escrow in a designated staking contract).

The initial stake distribution corresponding to a stake allocation function. Corresponding to each stake allocation function S is an *initial distribution*, denoted S^*, which is given to every player as an input. This distribution allocates a positive integer amount of stake to each of a finite and non-zero number of identifiers, and can be thought of as chosen by an adversary, subject to any constraints imposed on the fraction of stake controlled by Byzantine players.

Stake allocation functions. If T is a sequence of transactions, then $\mathsf{S}(\mathsf{S}^*, \mathsf{T}, id)$ $(\in \mathbb{N})$ is the stake owned by identifier id after execution of the transactions in T. It will also be notationally convenient to let $\mathsf{S}(\mathsf{S}^*, \mathsf{T})$ denote the function which on input id gives output $\mathsf{S}(\mathsf{S}^*, \mathsf{T}, id)$. If T is a sequence of transactions, then $\mathsf{T} * \mathtt{tr}$ denotes the sequence T concatenated with $\mathtt{tr}$.

[16]For convenience, in the description of the PosT protocol in the proof of Theorem 5.1, we also allow players to issue and sign special types of transactions, for example to signal the end of an "epoch."

We conclude this section with some baseline assumptions about how stake works.[17] We assume that players' initial allocations can be transferred. Formally, given stake allocation functions $\{\mathsf{S}_1, \ldots, \mathsf{S}_j\}$, we assume that: for all initial distributions $\mathsf{S}_1^*, \ldots, \mathsf{S}_j^*$ and every finite subset I of identifiers, there exists a set of transactions T such that, no matter how they are ordered, $\mathsf{S}_h(\mathsf{S}_h^*, \mathsf{T}, id) = 0$ for every $h \in [j]$ and $id \in I$.[18] We also assume that some transactions $\mathtt{tr}$ are *benign* in the sense that they do not destroy this property: for every $\mathsf{S}_1^*, \ldots, \mathsf{S}_j^*$ and I, there exists T such that, no matter how the transactions of T are ordered, $\mathsf{S}_h(\mathsf{S}_h^*, \mathtt{tr} * \mathsf{T}, id) = 0$ for every $h \in [j]$ and $id \in I$.[19] We do not assume that all transactions are benign in this sense; for example, the PosT protocol that we describe for the proof of Theorem 5.1 uses (non-benign) transactions that are "certificates of guilt" which have the effect of freezing the assests of the implicated identifiers.

B.5 Blockchain protocol requirements

Confirmed transactions. Each *blockchain protocol* specifies a *confirmation rule* C, which is a function that takes as input an arbitrary set of messages M and returns a sequence T of the transactions among those messages. At timeslot t, if M is the set of all messages received by an honest player p at timeslots $\leq t$, then p regards the transactions in $C(M)$ as *confirmed.* For a set of messages M, define $\mathsf{S}(\mathsf{S}^*, M, id) := \mathsf{S}(\mathsf{S}^*, C(M), id)$ and $\mathsf{S}(\mathsf{S}^*, M) := \mathsf{S}(\mathsf{S}^*, C(M))$.

The requirements on a blockchain protocol are that it should be *live* and *consistent.* For the sake of simplicity (to avoid the discussion of small error probabilities), in this paper we consider versions of liveness and consistency that apply to deterministic protocols. The proofs of our negative results (Theorems 4.1 and 4.2) will apply directly to deterministic protocols, but are easily extended to give analogous impossibility results for probabilistic protocols, simply by accounting for the appropriate error probabilities. The proof of our positive result (Theorem 5.1) uses a deterministic protocol, which only strengthens the result.

Defining liveness. We say a protocol is *live* if there exists a constant T_l, which may depend on the message delay bound Δ and the other determined protocol inputs, such that, whenever the environment sends a transaction $\mathtt{tr}$ to an honest player at some timeslot t, $\mathtt{tr}$ is among the sequence of confirmed transactions for every active honest player at every timeslot $t' \geq \max\{t, \mathrm{GST}\} + T_l$. (In the synchronous model, GST should be interpreted as 0.)

Defining consistency. Suppose the sequence of confirmed transactions for p at t is $\sigma = (\mathtt{tr}_1, \ldots, \mathtt{tr}_k)$, and that the sequence of confirmed transactions for p' at $t' \geq t$ is $\sigma' = (\mathtt{tr}'_1, \ldots, \mathtt{tr}'_{k'})$. We say a blockchain protocol is *consistent* if it holds in all executions that, whenever p and p' are honest:

- If $p = p'$ then σ' *extends* σ, meaning that $k' \geq k$ and $\mathtt{tr}_i = \mathtt{tr}'_i$ for each $i \in [1, k]$.
- Either σ extends σ', or σ' extends σ.

B.6 Protocols, executions, and ρ-bounded adversaries

Specifying blockchain protocols and executions. A blockchain protocol is a tuple $(\Sigma, \mathcal{O}, C, \mathcal{S})$, where Σ is the state machine diagram determining honest players, $\mathcal{O}$ is a set of oracles (some of which may be permitters), C is a confirmation rule, and $\mathcal{S} = \{\mathsf{S}_1, \ldots, \mathsf{S}_j\}$ is a set of stake allocation functions. An *execution* of the protocol $(\Sigma, \mathcal{O}, C, \mathcal{S})$ is a specification of the set of players $\mathcal{P}$, the

[17]These are important only in the proof of Theorem 4.2. They hold for the PosT protocol described in Section 5.1, and would presumably be satisfied by any reasonable PoS protocol.

[18]For example, if a stake allocation function represents native cryptocurrency, T could comprise payments transferring all stake initially owned by identifiers in I to identifiers outside of I. If the stake allocation function tracks stake-in-escrow, T could include one unstaking transaction for each $id \in I$.

[19]For example, a simple payment between two identifiers outside of I would presumably be a benign transaction.

player allocation, the state diagram of each player and their inputs, and the following values for each player p at each timeslot: (i) p's state at the beginning of the timeslot; (ii) the multiset of messages received by p; (iii) the oracle queries sent by p; (iv) the oracle responses received by p; (v) the messages disseminated by p.

Defining ρ-bounded adversaries. We say that an execution of a protocol is *ρ-bounded* if:

- The adversary is externally ρ-bounded (in the sense of Section B.3).
- For each stake allocation function $\mathsf{S} \in \mathcal{S}$, and among active players, Byzantine players never control more than a ρ fraction of the stake. Formally, for every honest player p at timeslot t, if T is the set of transactions that are confirmed for p at t in this execution, then at most a ρ fraction of the stake allocated to players active at t by $\mathsf{S}(\mathsf{S}^*, \mathsf{T})$ is allocated to Byzantine players.

When we say that "the adversary is ρ-bounded," we mean that we restrict attention to ρ-bounded executions. We say that a protocol is *ρ-resilient* when it it live and consistent under the assumption that the adversary is ρ-bounded. We say that a protocol is *ρ-resilient for liveness* when it it live under the assumption that the adversary is ρ-bounded, and that it is *ρ-resilient for consistency* when it is consistent under the assumption that the adversary is ρ-bounded.

B.7 The dynamically available and quasi-permissionless settings

Lewis-Pye and Roughgarden [Lewis-Pye and Roughgarden, 2023] describe a "degree of permissionlessness" hierarchy that parameterizes what a protocol may assume about the activity of honest players. This hierarchy is defined by four settings.[20] The two settings most relevant to this paper are:

(1) *Dynamically available setting.* At each moment in time, the protocol is aware of a dynamically evolving list of identifiers (e.g., public keys that currently have stake committed in a designated staking contract). The protocol may assume that at least *some* honest members of this list are active and participating in the protocol, but must function even when levels of participation fluctuate wildly. Proof-of-stake longest-chain protocols such as Ouroboros [Kiayias et al., 2017] and Snow White [Daian et al., 2019] are designed to function correctly in this setting.[21]

(2) *Quasi-permissionless setting.* At each moment in time, the protocol is aware of a dynamically evolving list of identifiers (as in the dynamically available setting), but now the protocol may assume that *all* honest members of the list are active. Proof-of-stake PBFT-style protocols such as Algorand [Chen and Micali, 2016] are typically interpreted as operating in this setting.

Formally defining the dynamically available setting. In the dynamically available setting, no assumptions are made about participation by honest players, other than the minimal assumption that, if any honest player owns a non-zero amount of stake, then at least one such player is active.[22]

Consider the protocol $(\Sigma, \mathcal{O}, \mathcal{C}, \mathcal{S})$. By definition, an execution of the protocol is *consistent with the dynamically available setting* if, for each stake allocation function $\mathsf{S} \in \mathcal{S}$:

[20]In order from "most permissionless" to "least permissionless" these are: the fully permissionless setting, the dynamically available setting, the quasi-permissionless setting and the permissioned setting. As the degree of permissionlessness of the setting increases—roughly, as the knowledge a protocol has about current participation decreases—positive results become more impressive and harder to come by (and impossibility results are increasingly to be expected).

[21]Proof-of-work longest-chain protocols such as Bitcoin [Nakamoto et al., 2008] function correctly even in the strictly more demanding "fully permissionless setting" defined in [Lewis-Pye and Roughgarden, 2023].

[22]Additional assumptions about the fraction of stake controlled by active honest players are phrased using the notion of ρ-bounded adversaries from Section B.6.

Whenever p is honest and active at t, with T the set of transactions confirmed for p at t in this execution, if there exists an honest player assigned a non-zero amount of stake by $\mathsf{S}(\mathsf{S}^*, \mathsf{T})$, then at least one such player is active at t.

Formally defining the quasi-permissionless setting. Consider the protocol $(\Sigma, \mathcal{O}, \mathcal{C}, \mathcal{S})$. By definition, an execution of the protocol is *consistent with the quasi-permissionless setting* if, for each stake allocation function $\mathsf{S} \in \mathcal{S}$:

Whenever p is honest and active at t, with T the set of transactions confirmed for p at t in this execution, every honest player that is assigned a non-zero amount of stake by $\mathsf{S}(\mathsf{S}^*, \mathsf{T})$ is active at t.

Thus, the quasi-permissionless setting insists on activity from every honest player that possesses any amount of any form of stake listed in the protocol description.

C Impossibility result for the dynamically available setting: The proof of Theorem 4.1

It suffices to prove the result for the synchronous model. (The following proof will be valid even if $\Delta = 1$.) Suppose the protocol $(\Sigma, \mathcal{O}, \mathcal{C}, \mathcal{S})$ is live (with liveness parameter T_l) and consistent in the dynamically available setting.

The intuition. Consider two disjoint sets of players X and Y that each own an equal amount of resources. Consider first an execution of the protocol in which players in X are Byzantine, while players in Y are honest. Players in X do not initially communicate with players in Y, but rather *simulate* between them an execution in which they are the only ones active. Because the protocol is live in the dynamically available setting, this simulated execution must produce a non-empty sequence T of confirmed transactions (even without contribution from the players in Y): Note that this conclusion would not hold if operating in the quasi-permissionless or permissioned settings, because the protocol might then require active participation from players owning a majority of resources to confirm transactions. Meanwhile, and for the same reason, the honest players in Y must eventually confirm a sequence of transaction T' that may be incompatible with T.

Now suppose that, at some later point, players in X disseminate all the messages that they would have disseminated if honest in their simulated execution. At this point, it is not possible for late arriving players to determine whether the players in X or the players in Y are honest. If $\alpha_B < x$, then $\alpha_H < x$ in a symmetrical execution, in which it is the players in X who are honest, while the players in Y are Byzantine and run their own simulated execution. (The notation α_H and α_B is defined in the discussion following Definition 3.1.)

The formal proof. We consider two non-empty sets of players, X and Y of equal size, and four protocol executions in which all parameters remain the same unless explicitly stated otherwise. For the sake of simplicity, we suppose all active players in $X \cup Y$ always hold a single unit of each external resource (if any), and all players in $X \cup Y$ begin with one unit of each form of stake (if any). If $\mathcal{S}$ is non-empty, then we suppose that, for each $\mathsf{S} \in \mathcal{S}$, the transactions $\mathtt{tr}_1$ and $\mathtt{tr}_2$ mentioned below do not affect the stake controlled by players in $X \cup Y$ (e.g., they are simple transfers between players outside of $X \cup Y$). If $\mathcal{S}$ is empty, then the form of these transactions is of no significance.

Execution 1. The only active players are those in X, who are active at all timeslots. The environment sends a single transaction $\mathtt{tr}_1$ to one of the players, p_1 say, at timeslot 1, and does not send any further transactions. All players are honest. By liveness, $\mathtt{tr}_1$ is confirmed for all active players by timeslot $1 + T_l$.

Execution 2. The only active players are those in Y, who are active at all timeslots. The environment sends a single transaction $\mathtt{tr}_2$ (with $\mathtt{tr}_2 \neq \mathtt{tr}_1$) to one of the players, p_2 say, at timeslot 1, and does not send any further transactions. All players are honest. By liveness, $\mathtt{tr}_2$ is confirmed for all active players by timeslot $1 + T_l$.

Execution 3. The active players are $X \cup Y$, and those players are active at all timeslots. Choose $t^* > 1+T_l$. The environment sends $\mathtt{tr}_1$ to p_1 and $\mathtt{tr}_2$ to p_2 at timeslot 1, and does not send any further transactions. Players in X are Byzantine, ignore messages from players in Y until t^*, and simulate the players in Execution 1 precisely (carrying out instructions as if they receive precisely the same messages at the same timeslots), except that they delay the dissemination of all messages until $t^* - 1$. At $t^* - 1$, players in X disseminate all messages that the players in Execution 1 disseminate at timeslots $< t^*$, and these messages are received by all active players by timeslot t^*. At timeslots $\geq t^*$ players in X act precisely like honest players, except that they pretend all messages received from players in Y at timeslots $< t^*$ were received at t^*. Players in Y are honest.

Execution 4. This is the same as Execution 3, but with the roles of X and Y reversed. The set of active players is $X \cup Y$, and those players are active at all timeslots. Timeslot t^* is defined as before. Again, the environment sends $\mathtt{tr}_1$ to p_1 and $\mathtt{tr}_2$ to p_2 at timeslot 1, and does not send any further transactions. Now players in Y are Byzantine, ignore messages from players in X until t^*, and simulate the players in Execution 2 precisely (carrying out instructions as if they receive precisely the same messages at the same timeslots), except that they delay the dissemination of all messages until $t^* - 1$. At $t^* - 1$, players in Y disseminate all messages that the players in Execution 2 disseminate at timeslots $< t^*$. At timeslots $\geq t^*$ players in Y act precisely like honest players, except that they pretend all messages received from players in X at timeslots $< t^*$ were received at t^*. Players in Y are honest.

Analysis. We first prove that at least one of Executions 3 and 4 must see a consistency violation. To see this, suppose towards a contradiction that Execution 3 does not. Note that, until t^*, Execution 3 is indistinguishable from Execution 2 as far as the honest players in Y are concerned, i.e. they receive precisely the same inputs, messages and oracle responses at each timeslot $< t^*$. It must therefore be the case that all players in Y regard $\mathtt{tr}_2$ as confirmed and $\mathtt{tr}_1$ as not confirmed (because it has not yet been received by those players) by timeslot $t^* - 1$. Let M^* be the set of messages received by all honest players by timeslot t^*. If there is no consistency violation then it must be the case that $C(M^*)$ is a sequence in which $\mathtt{tr}_1$ does not precede $\mathtt{tr}_2$.

In this case, consider Execution 4. Note that, until t^*, Execution 4 is indistinguishable from Execution 1 as far as the honest players in X are concerned, i.e. they receive precisely the same inputs, messages and oracle responses at each timeslot $< t^*$. It must therefore be the case that all players in X regard $\mathtt{tr}_1$ as confirmed and $\mathtt{tr}_2$ as not confirmed by timeslot $t^* - 1$. Note next, however, that the set of messages received by all honest players by timeslot t^* is precisely the same set M^* in Executions 3 and 4. We concluded above that $\mathtt{tr}_1$ does not precede $\mathtt{tr}_2$ in $C(M^*)$, meaning that Execution 4 sees a consistency violation at t^*.

To complete the proof, without loss of generality, suppose Execution 3 sees a consistency violation; the case that Execution 4 sees a consistency violation is symmetric. Towards a contradiction, suppose the protocol is EAAC with respect to the investment functions $\{\mathsf{R}_i(\cdot,\cdot)\}_{i\in[k]}$ and valuation function v. In that case, we must have $\alpha_B < 1$ at some timeslot $t_f > t^*$ in Execution 3 (and $\alpha_H \geq 1$ for all t). In that case, however, consider Execution 4. Because the set of messages received by each player by time t_f is the same in both executions, $R_i(p, t_f)$ is also the same for every $i \in [k]$ and $p \in X \cup Y$. It follows that the valuation v is also the same (for both X and Y) at time t_f in the two executions, and hence $\alpha_H < 1$ when evaluated at $t = t_f$ in Execution 4. This gives the required contradiction (violating (1) from Definition 3.1).