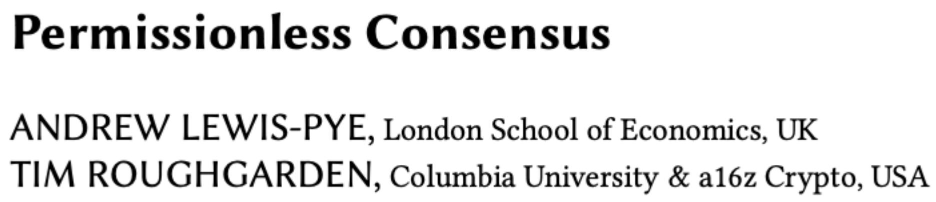
Meta-thread summarizing (most of) the results from my "Permissionless Consensus" paper with @AndrewLewisPye (1/11)

Intro thread: https://x.com/Tim_Roughgarden/status/1725205913953345995 (2/11)

.@AndrewLewisPye
 and I have, for years, been iterating on mathematical models for the design and analysis of permissionless consensus protocols 👇

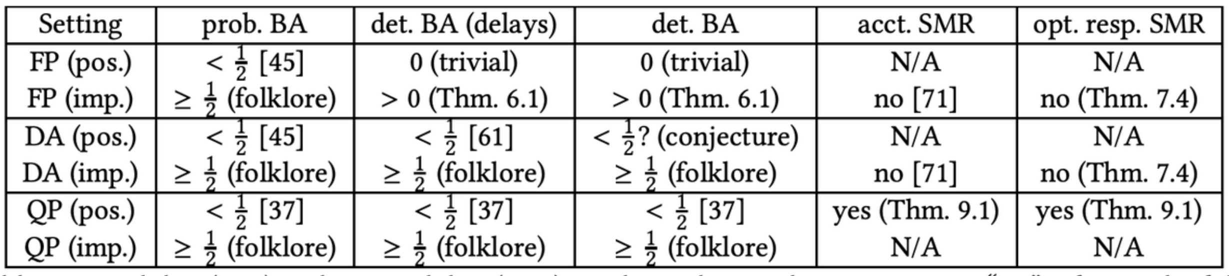


Permissionless Consensus

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

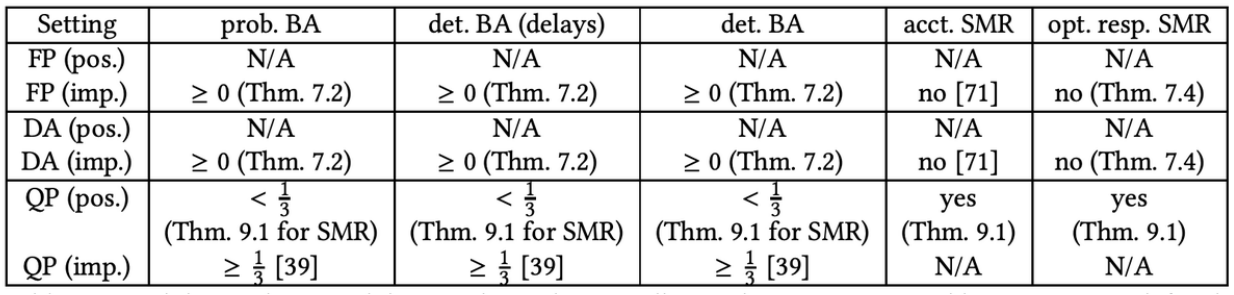
We're finally happy with one, a "sweet spot" model that is both very general (accommodating most of the major approaches to sybil-resistance, consensus, long-range attack defense, etc.) and user-friendly enough to enable lots of different possibility and impossibility results

For example (in synchrony):



Setting	prob. BA	det. BA (delays)	det. BA	acct. SMR	opt. resp. SMR
FP (pos.)	$< \frac{1}{2}$ [45]	0 (trivial)	0 (trivial)	N/A	N/A
FP (imp.)	$\geq \frac{1}{2}$ (folklore)	> 0 (Thm. 6.1)	> 0 (Thm. 6.1)	no [71]	no (Thm. 7.4)
DA (pos.)	$< \frac{1}{2}$ [45]	$< \frac{1}{2}$ [61]	$< \frac{1}{2}$? (conjecture)	N/A	N/A
DA (imp.)	$\geq \frac{1}{2}$ (folklore)	$\geq \frac{1}{2}$ (folklore)	$\geq \frac{1}{2}$ (folklore)	no [71]	no (Thm. 7.4)
QP (pos.)	$< \frac{1}{2}$ [37]	$< \frac{1}{2}$ [37]	$< \frac{1}{2}$ [37]	yes (Thm. 9.1)	yes (Thm. 9.1)
QP (imp.)	$\geq \frac{1}{2}$ (folklore)	$\geq \frac{1}{2}$ (folklore)	$\geq \frac{1}{2}$ (folklore)	N/A	N/A

And (in partial synchrony):



Setting	prob. BA	det. BA (delays)	det. BA	acct. SMR	opt. resp. SMR
FP (pos.)	N/A	N/A	N/A	N/A	N/A
FP (imp.)	≥ 0 (Thm. 7.2)	≥ 0 (Thm. 7.2)	≥ 0 (Thm. 7.2)	no [71]	no (Thm. 7.4)
DA (pos.)	N/A	N/A	N/A	N/A	N/A
DA (imp.)	≥ 0 (Thm. 7.2)	≥ 0 (Thm. 7.2)	≥ 0 (Thm. 7.2)	no [71]	no (Thm. 7.4)
QP (pos.)	$< \frac{1}{3}$ (Thm. 9.1 for SMR)	$< \frac{1}{3}$ (Thm. 9.1 for SMR)	$< \frac{1}{3}$ (Thm. 9.1 for SMR)	yes (Thm. 9.1)	yes (Thm. 9.1)
QP (imp.)	$\geq \frac{1}{3}$ [39]	$\geq \frac{1}{3}$ [39]	$\geq \frac{1}{3}$ [39]	N/A	N/A

There's no way I can reduce the paper down to a single tweetstorm (it's the longest research paper I've ever co-authored), so look for follow-up posts in the coming days that elaborate on our model and results

Full paper is at https://arxiv.org/pdf/2304.14701.pdf
(questions and comments welcome, as always!)

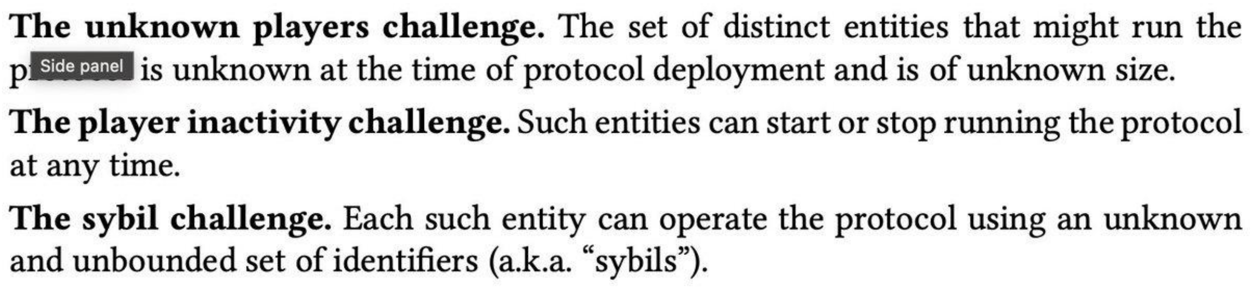
What is "permissionless consensus"?
https://x.com/Tim_Roughgarden/status/1725594927034179849 (3/11)

A quick thread on what @AndrewLewisPye and I mean by "permissionless consensus" (1/9)

The Bitcoin protocol solves a classical consensus problem known as "state machine replication (SMR)." In Bitcoin, the "state" is basically the current set of UTXOs. Researchers in distributed computing/systems have worked on SMR protocols since at least the 1980s (2/9)

So what's novel about Bitcoin's consensus protocol ("Nakamoto consensus") is not the problem it solves (SMR), but the very weak assumptions under which it solves it. Namely, the protocol assumes nothing about who might be running it (3/9)

In our paper, we formalize this idea via three distinct challenges that Nakamoto consensus overcomes: (4/9)



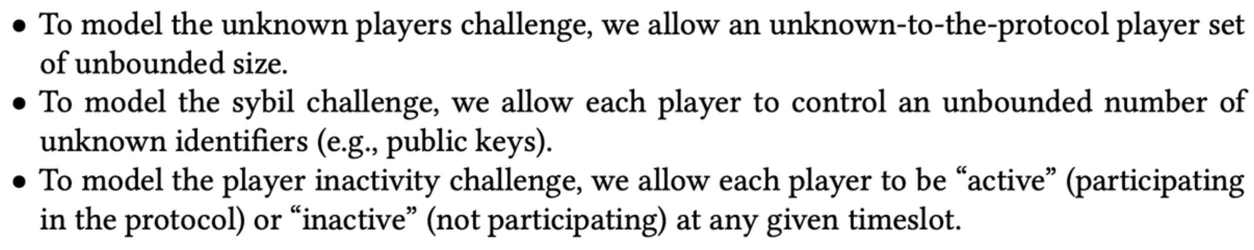
The unknown players challenge. The set of distinct entities that might run the p Side panel is unknown at the time of protocol deployment and is of unknown size.

The player inactivity challenge. Such entities can start or stop running the protocol at any time.

The sybil challenge. Each such entity can operate the protocol using an unknown and unbounded set of identifiers (a.k.a. "sybils").

Every subset of these challenges defines a distinct model, with its own set of possibility and impossibility results for reaching consensus (5/9)

All three challenges are built in to our model from the start: (6/9)



- To model the unknown players challenge, we allow an unknown-to-the-protocol player set of unbounded size.
- To model the sybil challenge, we allow each player to control an unbounded number of unknown identifiers (e.g., public keys).
- To model the player inactivity challenge, we allow each player to be "active" (participating in the protocol) or "inactive" (not participating) at any given timeslot.

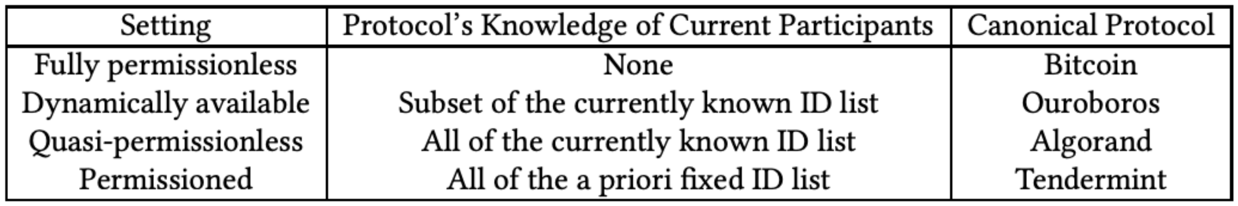
So by a "permissionless consensus protocol," we mean one that functions correctly (e.g., satisfies safety and liveness) even in the presence of one or more of these challenges (7/9)

Next up: a hierarchy of models, ordered according to the "degree of permissionlessness" (in the spirit of the classical hierarchies of fault models and message delay assumptions) (8/9)

Full paper is here: https://arxiv.org/pdf/2304.14701.pdf (9/9)

A hierarchy of "degrees of permissionlessness": https://x.com/Tim_Roughgarden/status/1729898971936526580 (4/11)

Can a proof-of-stake protocol be "truly permissionless"?
What does this question even mean?
A thread on the "hierarchy of permissionless" proposed by
@AndrewLewisPye and I:



Setting	Protocol's Knowledge of Current Participants	Canonical Protocol
Fully permissionless	None	Bitcoin
Dynamically available	Subset of the currently known ID list	Ouroboros
Quasi-permissionless	All of the currently known ID list	Algorand
Permissioned	All of the a priori fixed ID list	Tendermint

Researchers needs ways to organize and measure progress. What makes a result "better than" the last one? Hierarchies of successively weaker assumptions are a good way to do this

E.g., the synchronous and asynchronous models are opposite ends of a “degree of asynchrony” hierarchy; crash and Byzantine faults bookend a “fault severity” hierarchy

The goal is then to figure out the weakest-possible sets of assumptions under which good consensus protocols exist

We propose a four-level hierarchy (Perm < QP < DA < FP) that parameterizes the amount of knowledge that a protocol has about who's running it, which can be used as a yardstick to measure progress in permissionless consensus

At one extreme is the permissioned setting (Perm) where the participants are hard-wired into the protocol (as in e.g. Tendermint Core)

At the other extreme, which we call the "fully permissionless (FP)" setting, the protocol knows literally nothing about who's running it (e.g., Bitcoin operates under this assumption)

A middle ground is where a protocol does not have advance knowledge about who will run it but does, in the moment, have some idea about the current participants

In our "quasi-permissionless (QP)" setting, all potential current participants (e.g., all current stakers in a PoS protocol) are assumed to participate. In our "dynamically available (DA)" setting, some but not necessarily all of them participate

PBFT-style PoS protocols can stall in the DA setting due to low participation, but can have lots of nice properties (e.g., finality) in the QP setting. Longest-chain PoS protocols continue to function in the DA setting but sacrifice finality and other properties in exchange

E.g., Ethereum is effectively aspiring toward the best of both worlds: the basic guarantees of a longest-chain protocol provided the DA assumptions hold, plus the extra benefits of a PBFT-style protocol should the stronger QP assumptions hold

Tl;dr: (from least to most permissionless)
Perm < QP < DA < FP

See future threads for results on how possibility/impossibility results fundamentally change as you vary the degree of permissionlessness

Full paper: https://arxiv.org/pdf/2304.14701.pdf

An FLP-type impossibility result for protocols in the fully permissionless (and synchronous) setting: https://x.com/Tim_Roughgarden/status/1734736543284121849 (5/11)

The first main result in the "Permissionless Consensus" paper with @AndrewLewisPye
 is an FLP-type impossibility result for Bitcoin-style protocols (even in synchrony). What does that mean, exactly? (1/20)

The Bitcoin protocol is remarkable in that it assumes literally nothing about who is running it at any given moment---in our terminology, the protocol works even in the fully permissionless (FP) setting (2/20)

By "works" I mean it can be used to solve the state machine replication (SMR) and Byzantine Agreement (BA) problems, with guaranteed safety and liveness (for background, see https://x.com/Tim_Roughgarden/status/1489738014368710662) (3/20)

The fine print: (i) Byzantine players must control less than half the hashrate (else, they completely control the protocol) (4/20)

(ii) message delays must be bounded (else, a network partition can lead to a big re-org) [this is what "in synchrony" means] (5/20)

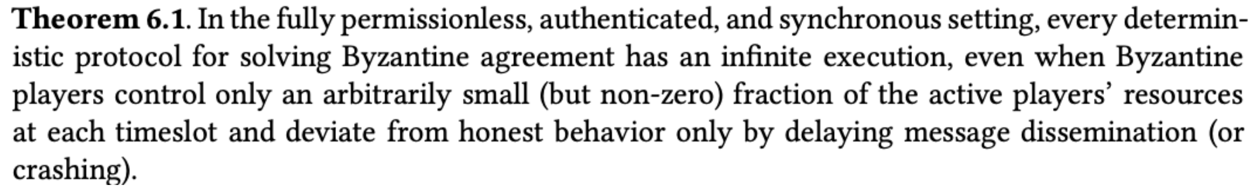
and (iii) the safety + liveness guarantees are only probabilistic (after enough time, even an attacker with a small amount of hashrate will get lucky enough to pull off a long re-org attack) (6/20)

Key question: to what extent are the drawbacks (i)-(iii) an artifact of Bitcoin's specific design? Maybe we can tweak the protocol somehow to fix one or more of them? (7/20)

Answer: *not at all*. (i)-(iii) are all fundamental compromises that every (arbitrarily clever) protocol must make in the FP setting (8/20)

Necessity of (i) is an easy exercise (and applies even to permissioned protocols). Necessity of (ii) will be addressed in a future thread, in a more general setting. This thread is about the unavoidability of probabilistic safety + liveness in the FP setting (9/20)

Here's the (semi-)formal statement of the impossibility result: (10/20)



Theorem 6.1. In the fully permissionless, authenticated, and synchronous setting, every deterministic protocol for solving Byzantine agreement has an infinite execution, even when Byzantine players control only an arbitrarily small (but non-zero) fraction of the active players' resources at each timeslot and deviate from honest behavior only by delaying message dissemination (or crashing).

The conclusion of the theorem---the impossibility of deterministic BA, even with a tiny number of benign faults---looks a lot like the FLP theorem, one of the most famous and influential impossibility results in distributed computing (see https://x.com/Tim_Roughgarden/status/1497349582560313348) (11/20)

The FLP impossibility result is driven by *asynchrony*, meaning unbounded message delays. It applies even to permissioned protocols (with an a priori known and unchanging set of participants) (12/20)

From 30K feet, the ambiguity any protocol must face in the asynchronous setting is: If you haven't heard from someone, is it because they've crashed, or because their messages simply haven't been delivered yet? (13/20)

But the new impossibility result quoted above holds in the *synchronous* setting, with bounded message delays. This result is driven not by asynchrony, but by the player inactivity and unknown players challenges in the FP setting (see https://x.com/Tim_Roughgarden/status/1725594930788049217) (14/20)

(Deterministic BA in the synchronous and permissioned setting is easy because one can use timeouts, as in the Dolev-Strong protocol, see https://x.com/Tim_Roughgarden/status/1489738014368710662) (15/20)

Thus, our result shows a sense in which (full) permissionlessness is as hard as asynchrony. In effect, a constantly changing player set (and deliberate unbounded message delays by a small number of Byzantine players) can substitute for an asynchronous network (16/20)

At a high level, the proof is a bivalency argument in the spirit of FLP, but the details are tricky (and it took us forever to get them right), for more info see Section 12 of the paper at https://arxiv.org/pdf/2304.14701.pdf (17/20)

Tl;dr 1: Bitcoin's probabilistic guarantees are an unavoidable consequence of its total lack of assumptions about who might be running it (18/20)

Tl;dr 2: Full permissionlessness is at least as hard as asynchrony (19/20)

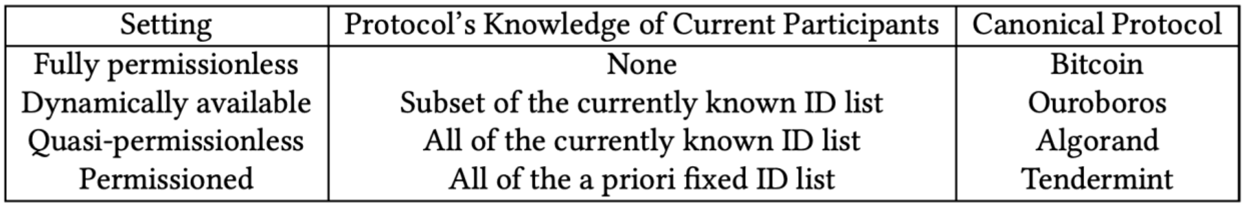
Next up: separating the FP and DA (dynamically available) settings! (20/20)

Separating the fully permissionless and dynamically available settings: https://x.com/Tim_Roughgarden/status/1735425682627285335 (6/11)

if you assume nothing about who's running a blockchain protocol (as in e.g. Bitcoin) => you're stuck with probabilistic safety and liveness guarantees (as in Bitcoin)

but what about under (slightly) stronger participation assumptions? (w/@AndrewLewisPye) (1/9)

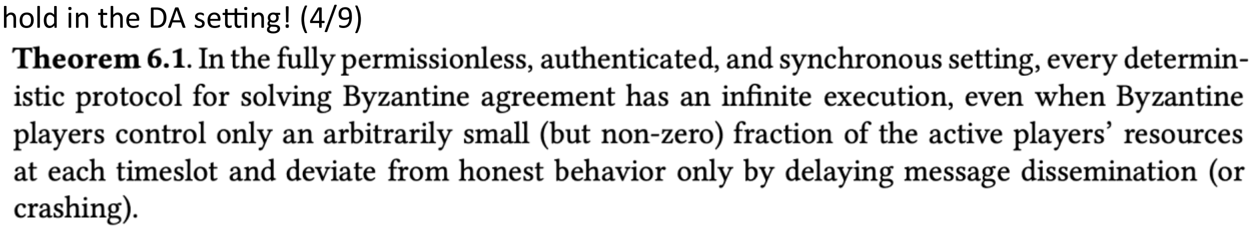
our dynamically available (DA) setting assumes something that our fully permissionless (FP) setting does not:
if there are any honest players with non-zero stake at some time t, then there is at least one such player that actually bothers to run the protocol at that time (2/9)



Setting	Protocol's Knowledge of Current Participants	Canonical Protocol
Fully permissionless	None	Bitcoin
Dynamically available	Subset of the currently known ID list	Ouroboros
Quasi-permissionless	All of the currently known ID list	Algorand
Permissioned	All of the a priori fixed ID list	Tendermint

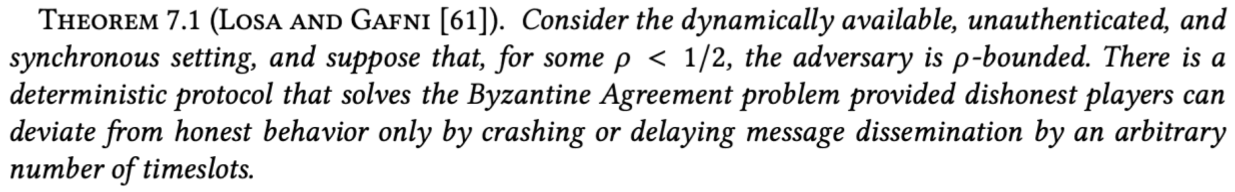
if that sounds like a pathetically weak extra assumption, you're right! it took us forever to figure out whether there's anything protocols can do in the DA setting that they can't already do in the FP setting (3/9)

eventually, we realized that a very cool and recent (deterministic) protocol due to Losa and Gafni can be adapted to show that our impossibility result for the FP setting (below) does not hold in the DA setting! (4/9)



Theorem 6.1. In the fully permissionless, authenticated, and synchronous setting, every deterministic protocol for solving Byzantine agreement has an infinite execution, even when Byzantine players control only an arbitrarily small (but non-zero) fraction of the active players' resources at each timeslot and deviate from honest behavior only by delaying message dissemination (or crashing).

here's the formal statement that, together with the impossibility result above, separates the FP and DA settings
["rho-bounded adversary" means that, at every time step, among active players, non-honest players control at most a rho fraction of the stake] (5/9)


Theorem 7.1 (Losa and Gafni [61]). *Consider the dynamically available, unauthenticated, and synchronous setting, and suppose that, for some $\rho < 1/2$, the adversary is ρ-bounded. There is a deterministic protocol that solves the Byzantine Agreement problem provided dishonest players can deviate from honest behavior only by crashing or delaying message dissemination by an arbitrary number of timeslots.*

how is the (super-weak) additional assumption in the DA setting useful? it allows a protocol to ignore, without suffering an immediate loss of liveness, all identifiers with no initial stake (6/9)

restricting attention to identifiers with non-zero initial stake has the effect of bounding the maximum number of identifiers that could be controlled by faulty players, opening the door to a Dolev-Strong-style approach (see https://x.com/Tim_Roughgarden/status/1489738014368710662 for background) (7/9)

if you're interested in the details, check out appendix C of the paper: https://arxiv.org/pdf/2304.14701.pdf (8/9)

so the DA setting is strictly easier than the FP setting. but not by much! coming up next thread: three unavoidable weaknesses of any (arbitrarily clever) blockchain protocol that makes only the participation assumptions of the DA setting (9/9)

Impossibility results for the dynamically available setting (for partial synchrony, for accountability, and for optimistic responsiveness): https://x.com/Tim_Roughgarden/status/1737843210830156112 (7/11)

Should proof-of-stake blockchain protocols follow the longest-chain tradition of Bitcoin, the PBFT-style tradition of permissioned protocols, or something else entirely? The trade-offs are pretty fascinating, so let's dive in (cc @AndrewLewisPye) (1/20)

The short answer is that PBFT-style PoS protocols make stronger assumptions about participation than longest-chain PoS protocols but, if these assumptions hold up, they can offer stronger guarantees like accountability, optimistic responsiveness, and partition-resilience (2/20)

The focus of this thread is the fundamental limitations of (PoW or PoS) longest-chain protocols; next thread will discuss the extra assumptions on participation that are necessary and sufficient for PBFT-style PoS protocols to overcome all of these limitations (3/20)

"Stronger assumptions about participation" is made precise using our "hierarchy of permissionlessness" https://x.com/Tim_Roughgarden/status/1729898971936526580 (4/20)

In the dynamically available (DA) setting, the only assumption about participation is that, at each time step, somebody with a non-zero amount of registered stake is bothering to run the protocol (5/20)

The good news about well-designed PoS longest-chain protocols is that they function correctly (i.e., solve state machine replication) even in the DA setting (assuming also that, at all time steps, a majority of the active stake is honest) (6/20)

The bad news is that *any* protocol that functions correctly in the DA setting (whether a PoS LC protocol or some other arbitrarily crazy protocol) provably cannot provide any of the following three desirable properties: (7/20)

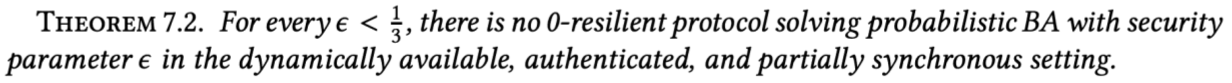
Property 1: Robustness to network partitions/periods of asynchrony. I.e., consistency always, and liveness whenever there's no partition/under normal network conditions (8/20)

Property 2: Accountability. I.e., if there's a consistency violation (like a double-spend), the protocol can automatically identify some validators to blame for it (think Ethereum slashing conditions) (9/20)

Property 3: Optimistic responsiveness, which is basically an instance-optimal version of liveness: latency (for transaction confirmation) should scale only with the realized message delays, which may be far smaller than the assumed worst-case bounds on message delays (10/20)

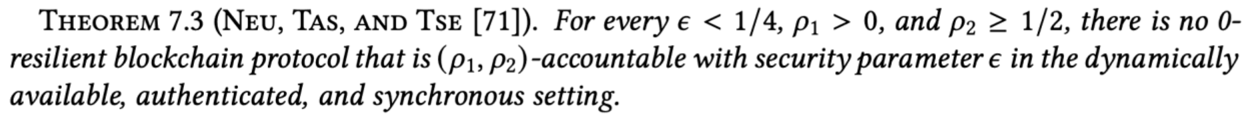
The fully permissionless (FP) setting in which PoW LC protocols like Bitcoin operate is even more challenging than the DA setting, so these three impossibility results apply directly to Bitcoin-like protocols (fulfilling an earlier promise)
https://x.com/Tim_Roughgarden/status/1734736555162435822 (11/20)

Formal statement of first impossibility result (about partial synchrony). Holds even if all players are honest (12/20)


Theorem 7.2. *For every $\epsilon < \frac{1}{3}$, there is no 0-resilient protocol solving probabilistic BA with security parameter ϵ in the dynamically available, authenticated, and partially synchronous setting.*

The logic of the proof resembles that of the CAP theorem from distributed systems, and rests on the ambiguity between unbounded network delays and unbounded periods of inactivity by honest players (13/20)

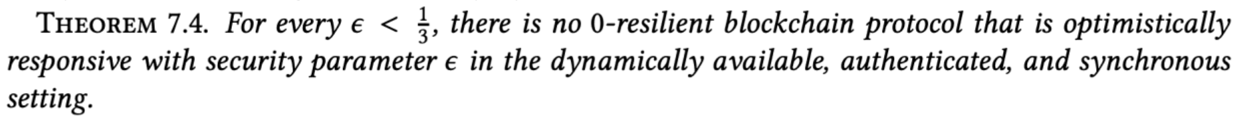
Formal statement of second impossibility result (about accountability), which follows directly from the excellent work of @jneu_net @ErtemTas @dntse (https://arxiv.org/abs/2105.06075). Holds even in the synchronous setting (14/20)



THEOREM 7.3 (NEU, TAS, AND TSE [71]). *For every $\epsilon < 1/4$, $\rho_1 > 0$, and $\rho_2 \geq 1/2$, there is no 0-resilient blockchain protocol that is (ρ_1, ρ_2)-accountable with security parameter ϵ in the dynamically available, authenticated, and synchronous setting.*

The weak assumptions of the DA setting drive the proof: (i) active honest players must confirm transactions even when the other players might plausibly be inactive; (ii) "late-arriving" players cannot disambiguate conflicting sets of allegedly confirmed transactions (15/20)

Formal statement of third impossibility result (about optimistic responsiveness). Holds even in the synchronous setting and with all players honest (16/20)



THEOREM 7.4. *For every $\epsilon < \frac{1}{3}$, there is no 0-resilient blockchain protocol that is optimistically responsive with security parameter ϵ in the dynamically available, authenticated, and synchronous setting.*

The logic of the proof is similar to that of the first impossibility result (about partial synchrony). Honest players can't figure out if the network is fast (messages delays=\delta) and their honest comrades are inactive or if the network is slow (message delays=\Delta) (17/20)

Tl;dr, a brutal but unavoidable trade-off: if your blockchain protocol is robust to fluctuating participation by honest players (i.e., functions correctly in the DA setting), then it cannot be optimistically responsive, accountable, or robust to network partitions (18/20)

Next thread: how to achieve optimistic responsiveness, accountability, and partition-resilience under the stronger participation assumptions of the quasi-permissionless setting (using a PoS PBFT-style approach) (19/20)

Full paper: https://arxiv.org/pdf/2304.14701.pdf (20/20)

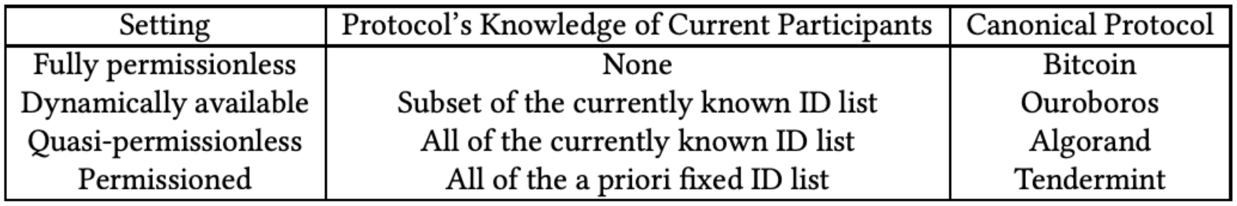
A possibility result for a proof-of-stake PBFT-style protocol in the quasi-permissionless setting: https://x.com/Tim_Roughgarden/status/1745124744431632875 (8/11)

Longest-chain protocols (or any protocol built for the dynamically available (DA) setting) must give up on a bunch of properties that you'd want (partition-resilience, accountability, optimistic responsiveness). Can PoS PBFT-style protocols do better? (1/10) [cc @AndrewLewisPye]

Tl;dr: yes, on all counts, at least if you're willing to adopt the stronger assumptions of the quasi-permissionless (QP) setting! (2/10)

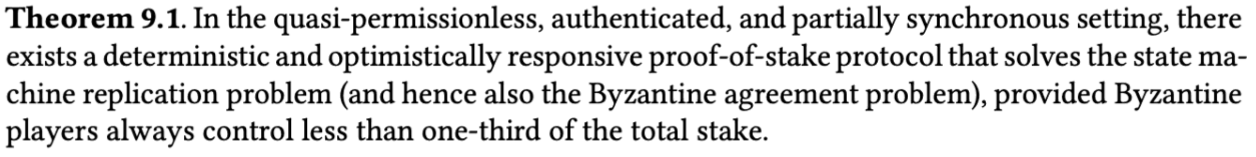
Quick review:
DA = some honest player with a non-zero amount of registered stake bothers to run the protocol

QP = *all* honest players with a non-zero amount of registered stake run the protocol (3/10)



Setting	Protocol's Knowledge of Current Participants	Canonical Protocol
Fully permissionless	None	Bitcoin
Dynamically available	Subset of the currently known ID list	Ouroboros
Quasi-permissionless	All of the currently known ID list	Algorand
Permissioned	All of the a priori fixed ID list	Tendermint

Here's our main result: (4/10)

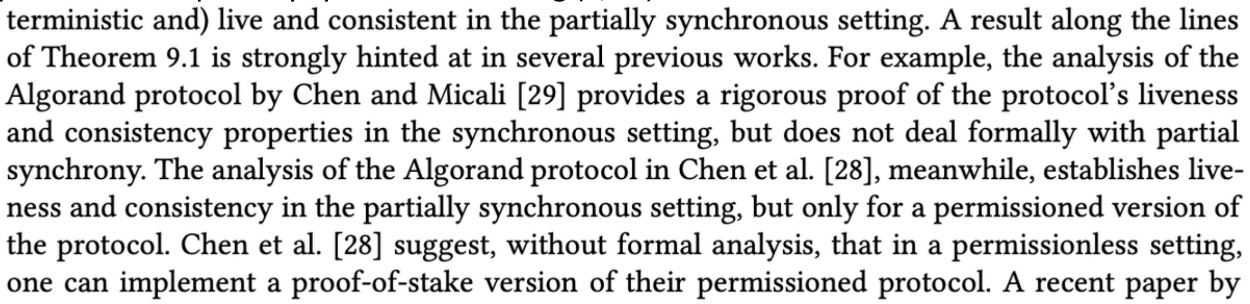


Theorem 9.1. In the quasi-permissionless, authenticated, and partially synchronous setting, there exists a deterministic and optimistically responsive proof-of-stake protocol that solves the state machine replication problem (and hence also the Byzantine agreement problem), provided Byzantine players always control less than one-third of the total stake.

This result shows that all of the impossibility results for the DA setting from the last thread (https://x.com/Tim_Roughgarden/status/1737843210830156112) can be overcome in the QP setting (5/10)

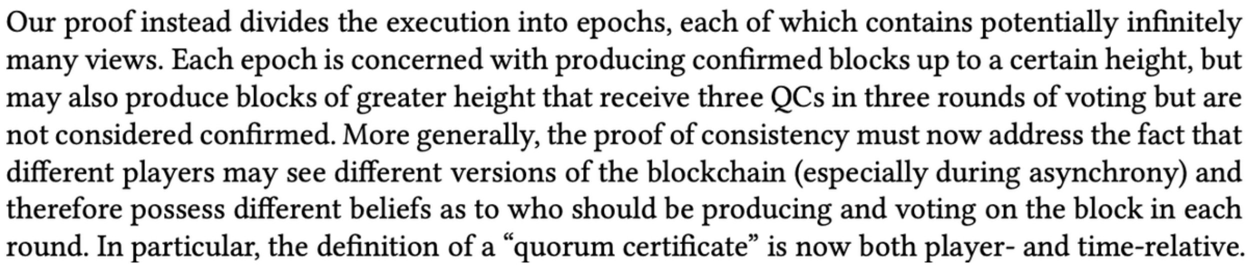
The protocol used in the proof can be viewed as an extension of the (permissioned) Hotstuff protocol (https://dl.acm.org/doi/pdf/10.1145/3293611.3331591) to a permissionless PoS protocol (6/10)

AFAWCT, this is the first complete proof of consistency and liveness for a permissionless protocol in the partially synchronous setting (7/10)



terministic and) live and consistent in the partially synchronous setting. A result along the lines of Theorem 9.1 is strongly hinted at in several previous works. For example, the analysis of the Algorand protocol by Chen and Micali [29] provides a rigorous proof of the protocol's liveness and consistency properties in the synchronous setting, but does not deal formally with partial synchrony. The analysis of the Algorand protocol in Chen et al. [28], meanwhile, establishes liveness and consistency in the partially synchronous setting, but only for a permissioned version of the protocol. Chen et al. [28] suggest, without formal analysis, that in a permissionless setting, one can implement a proof-of-stake version of their permissioned protocol. A recent paper by

For the consensus protocol nerds out there curious about the challenges involved: (8/10)



Our proof instead divides the execution into epochs, each of which contains potentially infinitely many views. Each epoch is concerned with producing confirmed blocks up to a certain height, but may also produce blocks of greater height that receive three QCs in three rounds of voting but are not considered confirmed. More generally, the proof of consistency must now address the fact that different players may see different versions of the blockchain (especially during asynchrony) and therefore possess different beliefs as to who should be producing and voting on the block in each round. In particular, the definition of a "quorum certificate" is now both player- and time-relative.

Next (and last) up: a thread on long-range attacks, and the relatively exotic cryptographic primitives that are necessary for in-protocol defenses against them (9/10)

Full paper is here: https://arxiv.org/pdf/2304.14701.pdf (10/10)

In-protocol defenses to long-range attacks require fancy cryptography: https://x.com/Tim_Roughgarden/status/1748471837468627356 (9/11)

Last thread (promise!) on results from my permissionless consensus paper with @AndrewLewisPye. The topic: are long-range attacks unavoidable for proof-of-stake blockchains? (1/11)

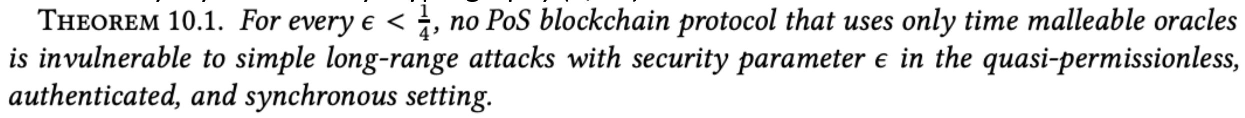
We've known about long-range attacks for a long time (e.g. https://blog.ethereum.org/2014/05/15/long-range-attacks-the-serious-problem-with-adaptive-proof-of-work). So what are they? (2/11)

Example:
1. let K denote the private keys controlling stake at block 10000
2. suppose by block 20000 owners of keys in K have cashed out and transferred all their stake to other users
3. an attacker buys the keys in K for a song (they're no longer valuable to their owners) (3/11)

4. the attacker can now fork the blockchain from block 10000 and unilaterally create as long a chain as it wants
Now if some new user shows up after the attack, how can they distinguish the attacker chain from the "true" chain? (4/11)

Long-range attacks are most commonly dealt with out-of-protocol (e.g. https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity). But can we repel them in-protocol, without appeal to something like "social consensus"? (5/11)

Answer: only if you use fancy cryptography (6/11)


THEOREM 10.1. *For every $\epsilon < \frac{1}{4}$, no PoS blockchain protocol that uses only time malleable oracles is invulnerable to simple long-range attacks with security parameter ϵ in the quasi-permissionless, authenticated, and synchronous setting.*

Here a "time malleable" cryptographic primitive is one that (i) can be evaluated quickly and (ii) whose output does not depend on the time of evaluation. E.g., typical signature schemes and cryptographic hash functions are time malleable in this sense (7/11)

Verifiable delay functions (VDFs), as seen e.g. in the Chia blockchain, are not time malleable because they fail property (i). They prevent the attacker from quickly creating an alternative history (as in step 4 above) (8/11)

Ephemeral keys, as seen e.g. in the Algorand blockchain, are not time malleable because they fail property (ii). Here, the old keys in K have expired and no longer enable an attacker to create new blocks (9/11)

The point of our result above is that fancy primitives like VDFs and ephemeral keys are the *only possible way* for proof-of-stake blockchains to defend against long-range attacks without any out-of-protocol assistance; there are no other options (10/11)

Full version of the paper is here: https://arxiv.org/pdf/2304.14701.pdf (11/11)

Full paper: https://arxiv.org/pdf/2304.14701.pdf (10/11)

Overview talk by @AndrewLewisPye: https://youtube.com/watch?v=dFYVckUzrIg (11/11)