To slash or not to slash?

New paper with @ebudish @AndrewLewisPye on provable slashing guarantees (both possibility and impossibility results), details below
1/25

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

Consider the bummer scenario in which an attacker acquires enough resources to interfere with your blockchain protocol (e.g., 51% of the hashrate of a Bitcoin-like protocol or 34% of the stake of an Ethereum-like protocol). Is it game over?
2/25

Ideally, a protocol would be able to "fight back" against an attacker that uses its power to violate the protocol's consistency (e.g., rolling back transactions to double-spend), and could do so without collateral damage to honest participants
3/25

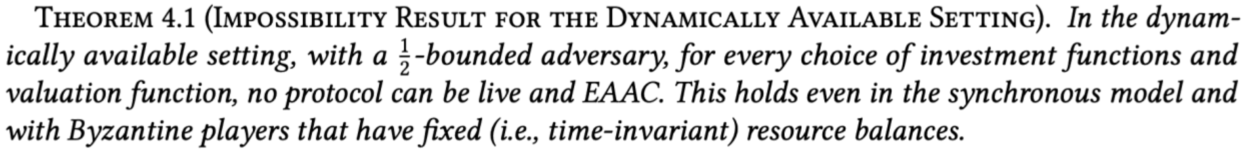
For example, this is one of the primary aspirations of slashing in a proof-of-stake protocol like Ethereum
4/25

We define the EAAC property (expensive to attack in the absence of collapse) to capture this idea of carrying out targeted punishment against an attacker responsible for a consistency violation (whether by PoS slashing or other means)
5/25

The goal of our paper is to map out fundamental possibility and impossibility results about EAAC protocols. Which types of protocols are capable of achieving the EAAC property, and under what assumptions?
6/25

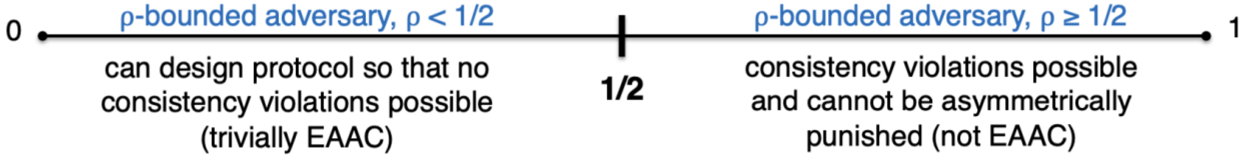
While our focus is on foundations rather than specific protocols, we highlight below several implications for practical blockchain protocol design (e.g., making precise the common belief that the merge has increased Ethereum’s economic security)
7/25

Result 1: You can't get EAAC in the dynamically available setting (in which non-malicious players might or might not be online): once an adversary is big enough to cause a consistency violation, it is big enough to evade targeted punishment
8/25



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.*

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)
9/25



0

ρ-bounded adversary, ρ < 1/2

can design protocol so that no consistency violations possible (trivially EAAC)

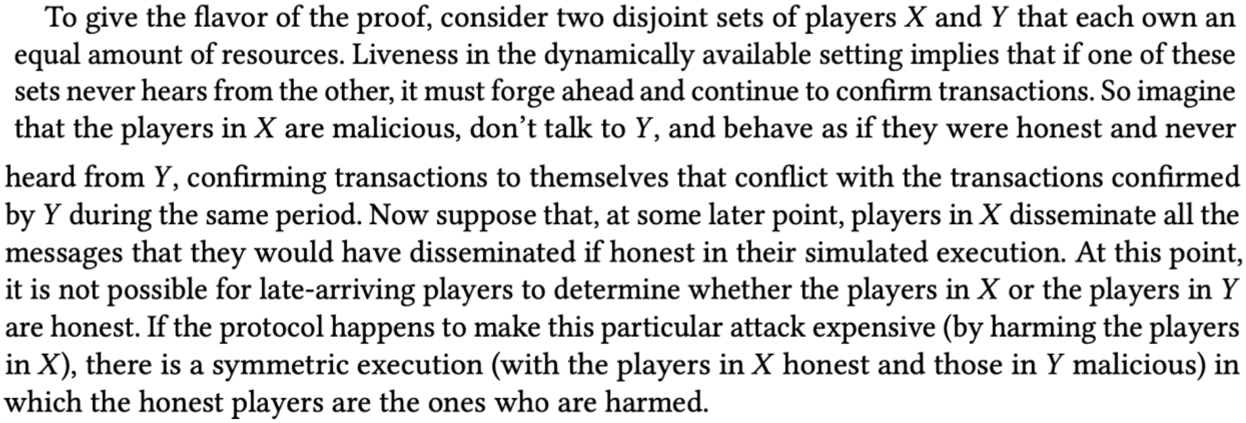
1/2

ρ-bounded adversary, ρ ≥ 1/2

consistency violations possible and cannot be asymmetrically punished (not EAAC)

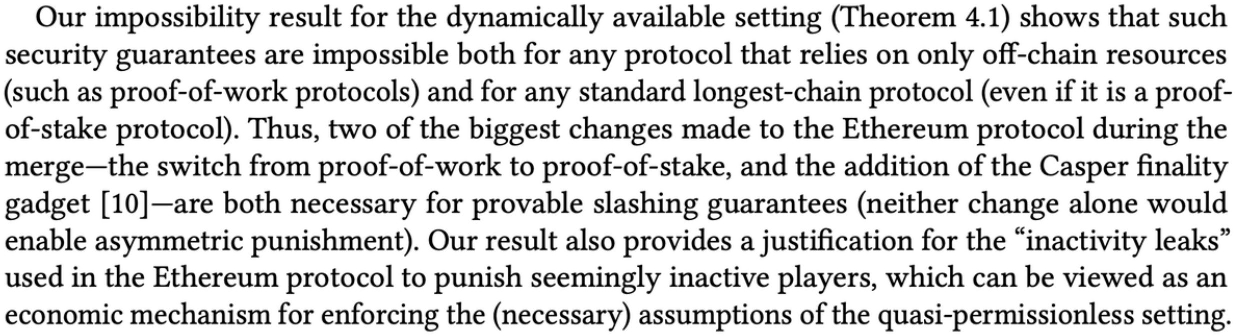
1

Intuition for proof:
10/25



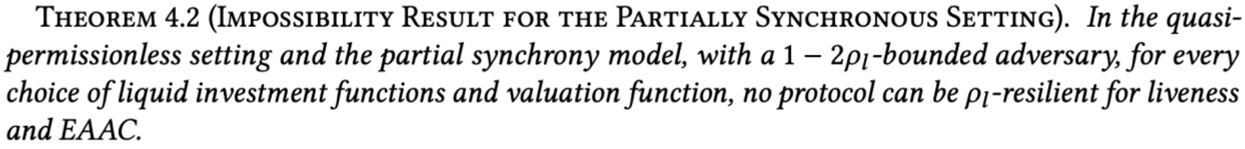
To give the flavor of the proof, consider two disjoint sets of players X and Y that each own an equal amount of resources. Liveness in the dynamically available setting implies that if one of these sets never hears from the other, it must forge ahead and continue to confirm transactions. So imagine that 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.

Implications for Ethereum and other PoS protocols:
11/25



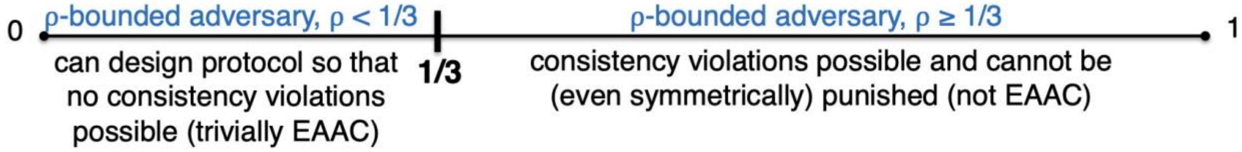
Our impossibility result for the dynamically available setting (Theorem 4.1) shows that such security guarantees are impossible both for any protocol that relies on only off-chain resources (such as proof-of-work protocols) and for any standard longest-chain protocol (even if it is a proof-of-stake protocol). 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 [10]—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.

Result 2: You can't get EAAC in the partially synchronous setting (in which the communication network may be periodically unreliable, e.g due to DoS attacks): once an adversary is big enough to cause a consistency violation, it is big enough to evade targeted punishment
12/25

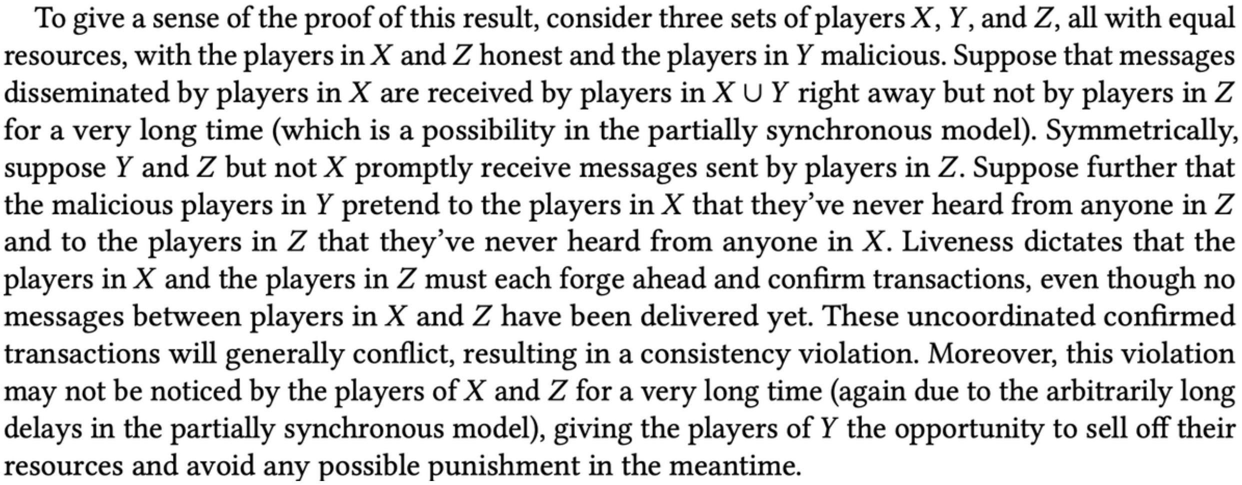


THEOREM 4.2 (IMPOSSIBILITY RESULT FOR THE PARTIALLY SYNCHRONOUS SETTING). *In the quasi-permissionless setting and the partial synchrony model, with a $1 - 2\rho_l$-bounded adversary, for every choice of liquid investment functions and valuation function, no protocol can be ρ_l-resilient for liveness and EAAC.*

In particular, this result implies that slashing in a proof-of-stake protocol cannot achieve its intended purpose if message delays cannot be bounded a priori
13/25

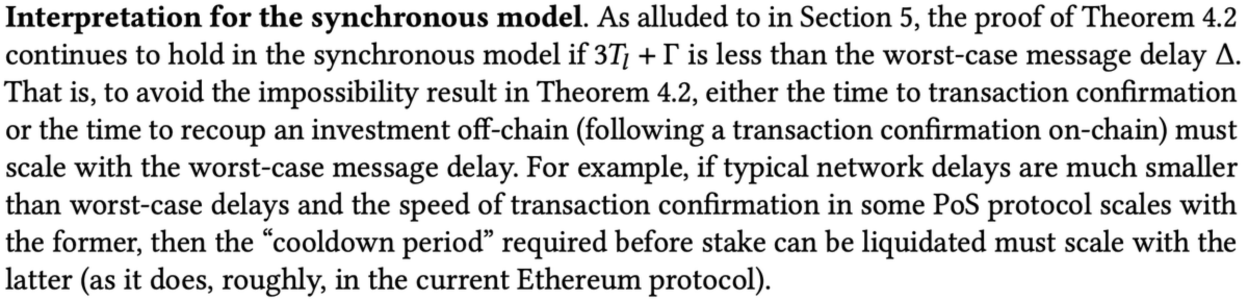

0
ρ-bounded adversary, ρ < 1/3
ρ-bounded adversary, ρ ≥ 1/3
1
can design protocol so that no consistency violations possible (trivially EAAC)
1/3
consistency violations possible and cannot be (even symmetrically) punished (not EAAC)

Intuition for proof:
14/25



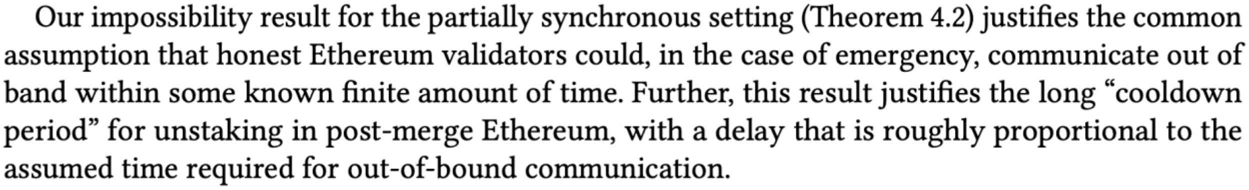
To give a sense of the proof of this result, consider three sets of players X, Y, and Z, all with equal resources, with the players in X and Z honest and the players in Y malicious. Suppose that messages disseminated by players in X are received by players in $X \cup Y$ right away but not by players 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.

Variation of result 2: even if there *is* a worst-case bound D* on message delay (which should apply even when the protocol is under attack), to have any hope of EAAC, withdraw delays (e.g., the cooldown period after unstaking) must be proportional to D*
15/25



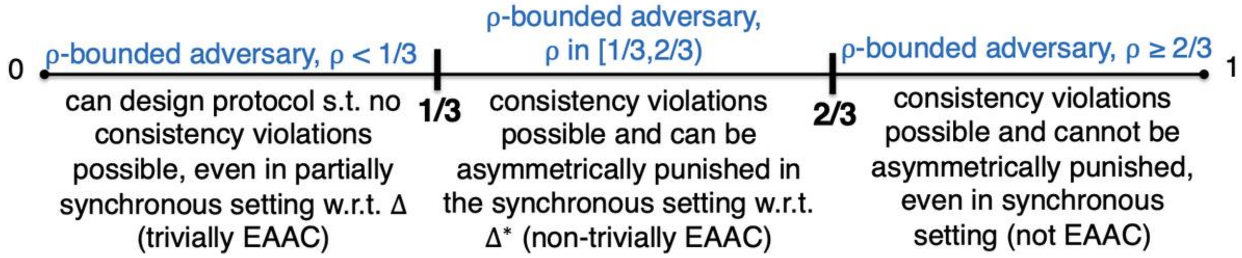
Interpretation for the synchronous model. As alluded to in Section 5, the proof of Theorem 4.2 continues to hold in the synchronous model if $3T_l + \Gamma$ is less than the worst-case message delay Δ. That is, to avoid the impossibility result in Theorem 4.2, either the time to transaction confirmation 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).

Implications for Ethereum and other PoS protocols:
16/25

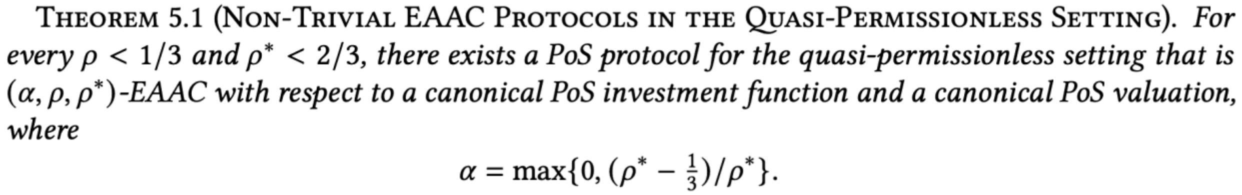

Our impossibility result for the partially synchronous setting (Theorem 4.2) justifies the common assumption that honest Ethereum validators could, in the case of emergency, communicate out of band within some known finite amount of time. Further, this result justifies the long "cooldown period" for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

Recap: to get EAAC, it's necessary to work in the quasi-permissionless setting (where honest players are assumed to be online) and have some worst-case bound on message delays. Are these assumptions also sufficient?
17/25

Answer: yes! EAAC is achievable by a PoS protocol with slashing, provided the attacker controls less than *two-thirds* of the overall stake
18/25

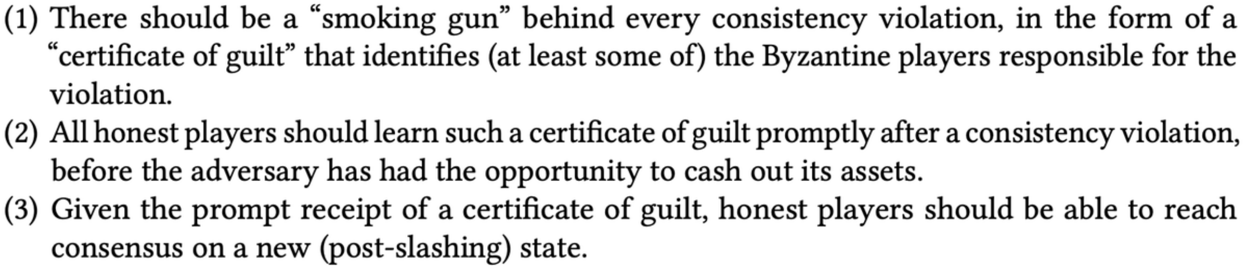

0
ρ-bounded adversary, ρ < 1/3
can design protocol s.t. no consistency violations possible, even in partially synchronous setting w.r.t. Δ (trivially EAAC)
1/3
ρ-bounded adversary, ρ in [1/3,2/3)
consistency violations possible and can be asymmetrically punished in the synchronous setting w.r.t. Δ* (non-trivially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

More generally, our protocol operates at a speed proportional to the typical message delay D (order of seconds) when there are no consistency violations and carries out slashing at a speed proportional to worst-case (perhaps out-of-band) message delay D* (order of days)
19/25


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*

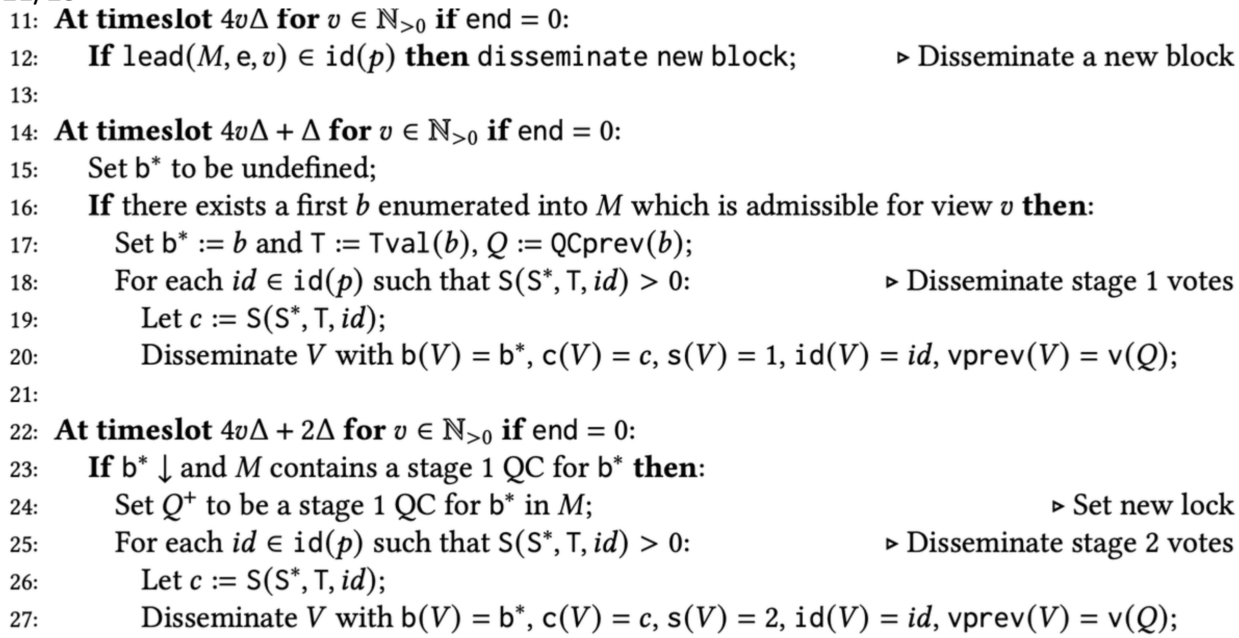
$$\alpha = \max\{0, (\rho^* - \tfrac{1}{3})/\rho^*\}.$$

How does the protocol work? Achieving EAAC requires addressing three challenges:
20/25


(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.

We address the first challenge by starting from a PBFT-style protocol that is accountable, namely Tendermint (which we extend to a quasi-permissionless PoS protocol)

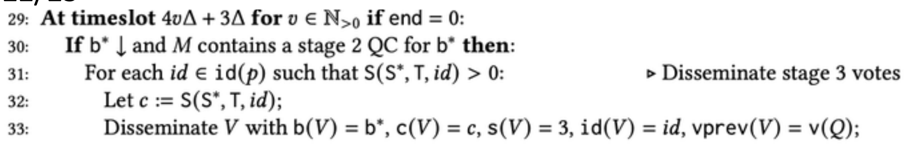
21/25



```
At timeslot $4v\Delta$ for $v \in \mathbb{N}_{>0}$ if end = 0:
    If lead$(M, \mathsf{e}, v) \in$ id$(p)$ then disseminate new block;        ▷ Disseminate a new block

At timeslot $4v\Delta + \Delta$ for $v \in \mathbb{N}_{>0}$ if end = 0:
    Set $\mathsf{b}^*$ to be undefined;
    If there exists a first $b$ enumerated into $M$ which is admissible for view $v$ then:
        Set $\mathsf{b}^* := b$ and $\mathsf{T} :=$ Tval$(b)$, $Q :=$ QCprev$(b)$;
        For each $id \in$ id$(p)$ such that $\mathsf{S}(\mathsf{S}^*, \mathsf{T}, id) > 0$:        ▷ Disseminate stage 1 votes
            Let $c := \mathsf{S}(\mathsf{S}^*, \mathsf{T}, id)$;
            Disseminate $V$ with $\mathsf{b}(V) = \mathsf{b}^*$, $\mathsf{c}(V) = c$, $\mathsf{s}(V) = 1$, $\mathsf{id}(V) = id$, $\mathsf{vprev}(V) = \mathsf{v}(Q)$;

At timeslot $4v\Delta + 2\Delta$ for $v \in \mathbb{N}_{>0}$ if end = 0:
    If $\mathsf{b}^* \downarrow$ and $M$ contains a stage 1 QC for $\mathsf{b}^*$ then:
        Set $Q^+$ to be a stage 1 QC for $\mathsf{b}^*$ in $M$;        ▷ Set new lock
        For each $id \in$ id$(p)$ such that $\mathsf{S}(\mathsf{S}^*, \mathsf{T}, id) > 0$:        ▷ Disseminate stage 2 votes
            Let $c := \mathsf{S}(\mathsf{S}^*, \mathsf{T}, id)$;
            Disseminate $V$ with $\mathsf{b}(V) = \mathsf{b}^*$, $\mathsf{c}(V) = c$, $\mathsf{s}(V) = 2$, $\mathsf{id}(V) = id$, $\mathsf{vprev}(V) = \mathsf{v}(Q)$;
```

We address the second challenge by adding a third stage of voting in each view (we prove that the usual two stages are inadequate)

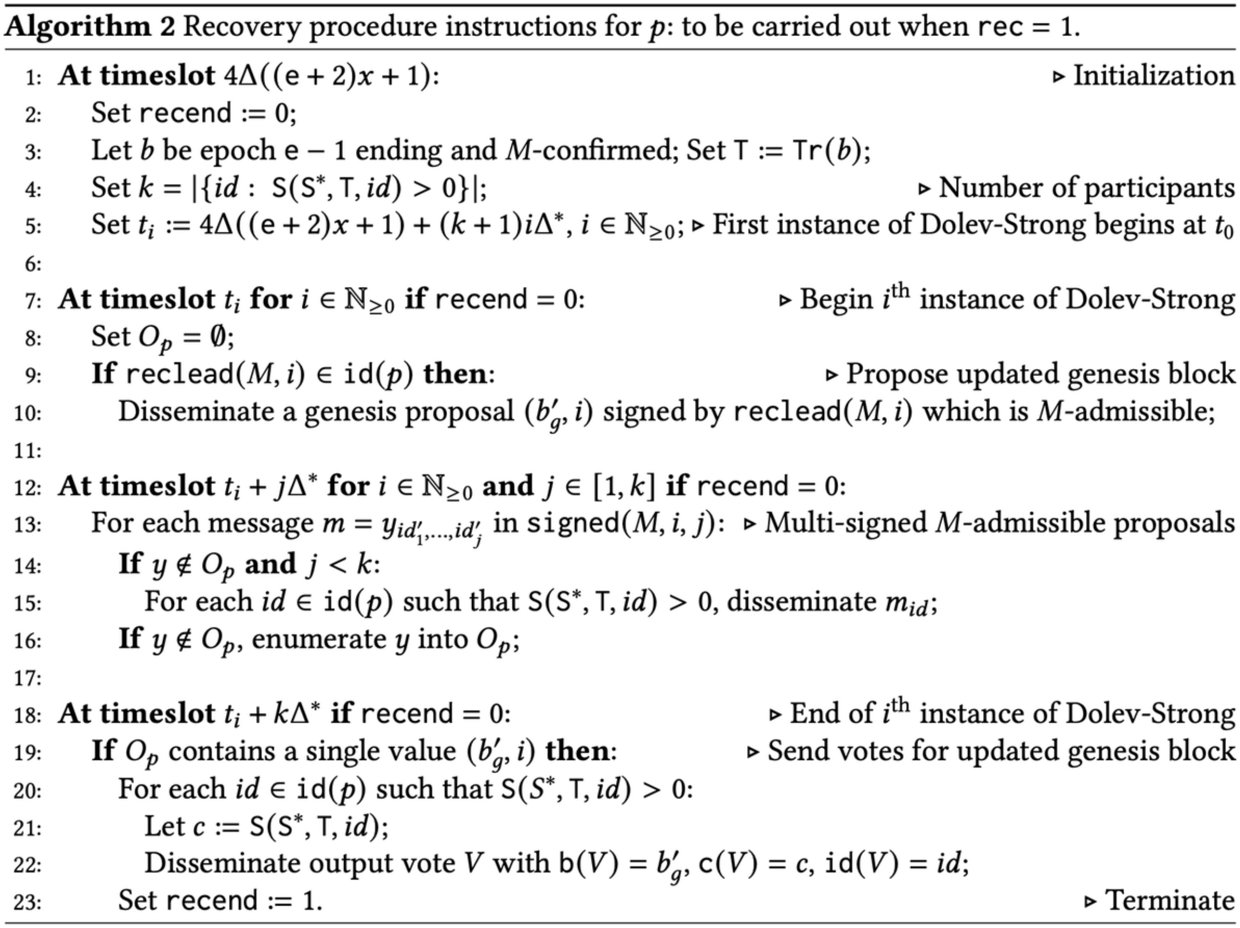
22/25



```
At timeslot $4v\Delta + 3\Delta$ for $v \in \mathbb{N}_{>0}$ if end = 0:
    If $\mathsf{b}^* \downarrow$ and $M$ contains a stage 2 QC for $\mathsf{b}^*$ then:
        For each $id \in$ id$(p)$ such that $\mathsf{S}(\mathsf{S}^*, \mathsf{T}, id) > 0$:        ▷ Disseminate stage 3 votes
            Let $c := \mathsf{S}(\mathsf{S}^*, \mathsf{T}, id)$;
            Disseminate $V$ with $\mathsf{b}(V) = \mathsf{b}^*$, $\mathsf{c}(V) = c$, $\mathsf{s}(V) = 3$, $\mathsf{id}(V) = id$, $\mathsf{vprev}(V) = \mathsf{v}(Q)$;
```

We address the third challenge by having 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

23/25

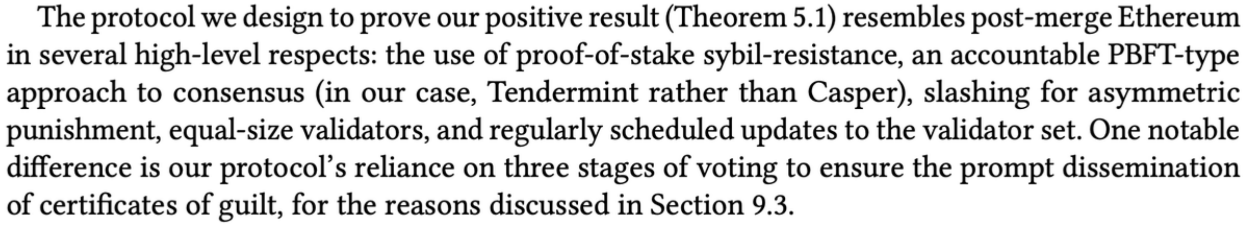


Algorithm 2 Recovery procedure instructions for p: to be carried out when $\texttt{rec} = 1$.

```
At timeslot $4\Delta((\mathsf{e}+2)x+1)$:                                          ▷ Initialization
    Set $\texttt{recend} := 0$;
    Let $b$ be epoch $\mathsf{e}-1$ ending and $M$-confirmed; Set $\mathsf{T} := \mathtt{Tr}(b)$;
    Set $k = |\{id : \mathsf{S}(\mathsf{S}^*, \mathsf{T}, id) > 0\}|$;                  ▷ Number of participants
    Set $t_i := 4\Delta((\mathsf{e}+2)x+1) + (k+1)i\Delta^*$, $i \in \mathbb{N}_{\geq 0}$; ▷ First instance of Dolev-Strong begins at $t_0$

At timeslot $t_i$ for $i \in \mathbb{N}_{\geq 0}$ if $\texttt{recend} = 0$:            ▷ Begin $i^{\text{th}}$ instance of Dolev-Strong
    Set $O_p = \emptyset$;
    If $\mathtt{reclead}(M, i) \in \mathtt{id}(p)$ then:                          ▷ Propose updated genesis block
        Disseminate a genesis proposal $(b'_g, i)$ signed by $\mathtt{reclead}(M, i)$ which is $M$-admissible;

At timeslot $t_i + j\Delta^*$ for $i \in \mathbb{N}_{\geq 0}$ and $j \in [1, k]$ if $\texttt{recend} = 0$:
    For each message $m = y_{id'_1,\ldots,id'_j}$ in $\mathtt{signed}(M, i, j)$:  ▷ Multi-signed $M$-admissible proposals
        If $y \notin O_p$ and $j < k$:
            For each $id \in \mathtt{id}(p)$ such that $\mathsf{S}(\mathsf{S}^*, \mathsf{T}, id) > 0$, disseminate $m_{id}$;
        If $y \notin O_p$, enumerate $y$ into $O_p$;

At timeslot $t_i + k\Delta^*$ if $\texttt{recend} = 0$:                            ▷ End of $i^{\text{th}}$ instance of Dolev-Strong
    If $O_p$ contains a single value $(b'_g, i)$ then:                      ▷ Send votes for updated genesis block
        For each $id \in \mathtt{id}(p)$ such that $\mathsf{S}(S^*, \mathsf{T}, id) > 0$:
            Let $c := \mathsf{S}(\mathsf{S}^*, \mathsf{T}, id)$;
            Disseminate output vote $V$ with $\mathsf{b}(V) = b'_g$, $\mathsf{c}(V) = c$, $\mathtt{id}(V) = id$;
        Set $\texttt{recend} := 1$.                                             ▷ Terminate
```

Comparison between our protocol and the Ethereum protocol:

24/25



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, for the reasons discussed in Section 9.3.

Link to the full paper: https://arxiv.org/abs/2405.09173

As always, questions and comments welcome!

25/25