Accountable Safety Implies Finality

Joachim Neu
Stanford University, USA

Ertem Nusret Tas
Stanford University, USA

David Tse
Stanford University, USA

Abstract

Motivated by proof-of-stake (PoS) blockchains such as Ethereum, two key desiderata have recently been studied for Byzantine-fault tolerant (BFT) state-machine replication (SMR) consensus protocols: *Finality* means that the protocol retains consistency, as long as less than a certain fraction of validators are malicious, even in *partially-synchronous* environments that allow for temporary violations of assumed network delay bounds. *Accountable safety* means that in any case of inconsistency, a certain fraction of validators can be *identified* to have *provably* violated the protocol. Earlier works have developed impossibility results and protocol constructions for these properties separately. We show that accountable safety implies finality, thereby unifying earlier results.

2012 ACM Subject Classification Security and privacy → Distributed systems security

Keywords and phrases Byzantine-fault tolerant (BFT), state-machine replication (SMR), consensus, network partition, safety, finality, accountability, accountable safety, forensics

Funding JN, ENT, and DT are supported by a gift from the Ethereum Foundation. JN is supported by the Protocol Labs PhD Fellowship and the Reed-Hodgson Stanford Graduate Fellowship. ENT is supported by the Stanford Center for Blockchain Research.

Acknowledgements We thank David Mazières and Srivatsan Sridhar for fruitful discussions.

1 Introduction

Consensus The purpose of a consensus protocol for state-machine replication (SMR) is for a set of parties to reach agreement on how to sequence incoming transactions into a linear order called a *ledger*. This task is non-trivial because communication between parties might be delayed, and some parties might deviate from the protocol in an arbitrary manner (*Byzantine faults*) with the goal to undermine consensus. A consensus protocol is *secure* if even in the presence of these disturbances, it guarantees two complementary properties: *safety*, meaning that the ledgers output by non-faulty parties across time are consistent ('bad events, *i.e.*, inconsistencies, never happen'), and *liveness*, meaning that transactions make it to the output ledger 'soon' ('good events, *i.e.*, transactions get confirmed, do happen frequently'). The protocol is then called Byzantine-fault tolerant (BFT), and the fraction of faulty parties it can tolerate while remaining secure is called its *resilience*.

Finality This basic formulation has subsequently been extended in two directions. On the one hand, while some early consensus protocols [9] assume that network communication *always* obeys a known delay upper-bound (*i.e.*, *synchronous network* [17]), later constructions [11, 5] pushed to strengthen security to ensure consistency also under temporary network delay-bound violations (*i.e.*, *partially-synchronous network* [11]). Such periods of asynchrony might

Authors listed alphabetically.

be caused, for instance, by temporary network partitions. The strengthened safety property that ensures consistency also under periods of asynchrony is called *finality* [3, 22].

Accountable Safety On the other hand, consensus protocols for proof-of-stake (PoS) blockchains such as Ethereum seek to strengthen safety to enable *accountability* [3, 4, 23, 22, 27, 20, 7, 2, 14, 15, 25]. In permissionless blockchains, parties are no longer inanimate computers which might exhibit technical faults but are otherwise aligned under the control of one organizational entity, but parties are instead controlled by different mutually distrusting self-interested players that might deviate from the protocol if they expect a profit from doing so. In this setting, it was proposed to strengthen safety to *accountable safety*, where besides ensuring consistency up to some adversarial resilience, a certain fraction of parties can also be identified to have provably violated the protocol in any case of a safety violation.

Prior Works Various works have studied the fundamental limits of finality [12, 11, 13] and of accountable safety [27], as well as their relationships to other desiderata such as liveness under dynamic participation [16, 23], and how protocols can be constructed that achieve various combinations of these properties. While finality and accountable safety 'feel similar', characterizing their exact relation has remained open. For instance, some protocols provide finality but not provide accountable safety [27, 21]. On the other hand, an additional round of voting to 'checkpoint' [19, 10] the output ledger of a consensus protocol designed for synchronous networks can also be used to upgrade that ledger to provide accountable safety, but does not yield a protocol for partially-synchronous networks (in particular, the modified protocol might never recover from a liveness fault induced during a period of asynchrony). Details are provided in Section 5.

Main Result We show that accountable safety implies finality. To this end, on a high level, we show that for any given protocol, if there exists an adversary strategy that leads to a safety violation under partial synchrony, then there exists an adversary strategy that leads to a safety violation but not enough adversary parties can be identified as protocol violators, even if the network is delay-free, *i.e.*, messages arrive instantly. Intuitively, the more constraints there are on network delays, the easier it is for a protocol to guarantee accountable safety. Our argument shows that even the weakest form of accountable safety, namely for delay-free networks, is still so strong that it implies finality.

The observation that accountable safety implies finality unifies prior works and directs future work: The availability–accountability dilemma [23] turns out to be implied by the availability–finality dilemma [22], a blockchain-variant of the CAP theorem [13, 16]. Impossibility results for theaccountability and liveness resiliences [27] turn out to be implied by impossibility results for consensus under partial synchrony [11]. Instead of designing finality gadgets [3, 26, 29, 8, 28, 18], it suffices to design accountability gadgets [23, 20].

Outline Section 2 introduces model and notation. Section 3 formally proves that accountable safety implies finality. Section 4 presents simplified alternative proofs of impossibility results about accountability from the newly proven implication and well-known impossibility results about finality. Section 5 elaborates on the implications and non-implications of our result for safety, liveness, and security under partial synchrony.

2 Model

2.1 Notation

Given a positive integer m, let $[m]$ denote the set $\{1, 2, \ldots, m\}$. An event is said to happen with probability *negligible* in the security parameter λ if its probability is $o\left(\frac{1}{\mathrm{poly}(\lambda)}\right)$.

2.2 Replicas and Clients

SMR consensus protocols have two types of participants: *replicas* and *clients*. Replicas are input *transactions* by the environment, interact with each other towards agreeing on how these transactions should be ordered, and make some protocol messages (*e.g.*, blocks, votes) available to clients upon request. *Clients* query replicas for these protocol messages, and, upon collecting messages from a sufficiently large subset of replicas, output a sequence of transactions called the *ledger* and denoted by LOG. The goal of the SMR protocol is to ensure that clients agree on a single ever-growing transaction sequence.

2.3 Environment and Adversary

Each of the n replicas has a (unique) cryptographic identity that is common knowledge to all replicas and clients. Up to f replicas can be corrupted at the beginning of the protocol execution by a computationally-bounded *adversary* $\mathcal{A}$, which then obtains the internal state of these replicas, and can make them deviate from the protocol in arbitrary ways (Byzantine faults). The remaining $(n-f)$ replicas are *honest* and follow the protocol as specified.

Time proceeds in slots. Replicas can exchange messages, subject to delays determined by the adversary. We consider a *partially-synchronous network* with adversary-environment tuple $(\mathcal{A}_{\mathrm{p}}, \mathcal{Z}_{\mathrm{p}})$, where the adversary can delay messages arbitrarily until a *global stabilization time* GST that can be chosen adaptively by the adversary. After GST, $\mathcal{A}_{\mathrm{p}}$ has to deliver messages within a delay upper-bound of Δ which is known to the protocol. If GST is known and zero, then the network is said to be *synchronous*, and denoted by $(\mathcal{A}_{\mathrm{s}}, \mathcal{Z}_{\mathrm{s}})$. Furthermore, a network is called *delay-free* and denoted by $(\mathcal{A}_{\mathrm{i}}, \mathcal{Z}_{\mathrm{i}})$ if all messages reach their recipients instantaneously, *i.e.*, the network is synchronous with $\Delta = 0$. The three network models are ordered in the sense that from partial synchrony via synchrony to delay-freeness, for fixed Δ, the adversary's capabilities are strictly increasingly constrained.

2.4 Safety and Liveness Resiliences

Let $\mathsf{LOG}_t^{\mathsf{cl}}$ denote the ledger in the view of a client cl at time slot t.

▶ **Definition 1.** *An SMR consensus protocol* Π *is* secure *with confirmation time* T_{confirm} *iff:*

- Safety: *For all* t, t' *and* $\mathsf{cl}, \mathsf{cl}'$*, either* $\mathsf{LOG}_t^{\mathsf{cl}}$ *is a prefix of* $\mathsf{LOG}_{t'}^{\mathsf{cl}'}$*, or vice versa.*
- Liveness: *If a transaction is input to an honest replica by some* t*, then, for any* $t' \geq \max(t, \mathsf{GST}) + T_{\mathrm{confirm}}$*, and all* cl*, the transaction is included in* $\mathsf{LOG}_{t'}^{\mathsf{cl}}$*.*

A protocol is said to provide *f-safety (f-liveness)* if the protocol satisfies safety (liveness), except with negligible probability, for any adversary controlling at most f replicas. Here, f is the protocol's safety (liveness) resilience.

▶ **Definition 2.** *An SMR consensus protocol satisfies* f-finality *(*i.e.*, is* f-final*) if it satisfies f-safety under a partially-synchronous network.*

Note that f-finality need not imply f-liveness after GST, *i.e.*, a protocol that is f-final might not be secure under a partially synchronous network due to liveness violations (cf. Section 5).

2.5 Accountable-Safety Resilience

Upon detecting inconsistent output ledgers, clients invoke a forensic protocol with the consensus messages they have received from the replicas and that have resulted in the

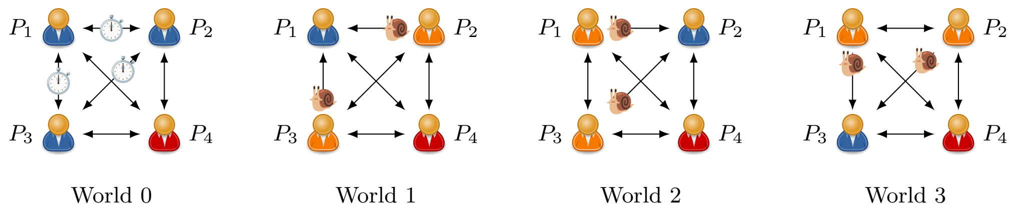

Figure 1 Execution of an SMR protocol with four replicas. World 0 has a partially-synchronous network, and worlds 1, 2 and 3 are delay-free. The red replica P_4 is adversary in all of the worlds. The orange replicas are adversary but do not violate the protocol rules other than delaying the sending/receiving of messages to/from the honest replica. Blue replicas are honest.

inconsistency. The forensic protocol outputs a cryptographic proof that identifies f_{a} replicas as protocol violators [27]. Building on the concept of α-*accountable-safety* [3, 20], the accountable-safety resilience is defined as follows:

▶ **Definition 3.** *An SMR consensus protocol provides accountable safety with resilience f_{a} (*i.e.*, is f_{a}-accountable safe) iff whenever there is a safety violation, except with negligible probability, (i) at least f_{a} adversarial replicas are identified by the forensic protocol as protocol violators, and (ii) no honest replica is identified.*

Note that the verdict of the forensic protocol is not conditional on assumptions such as a fraction of replicas being honest. However, we assume that the forensic protocol does know whether the network is partially-synchronous, synchronous, or delay-free. For instance, in a delay-free network, the forensic protocol is assumed to *know* that every message sent by an honest replica is instantly delivered to its recipients.

3 Accountability Implies Finality

By Definition 3, if a protocol provides $(f+1)$-accountable safety under partial synchrony (which is the strongest form of accountable safety, considering that among the models considered here, the adversary's capabilities are least constrained in the partially-synchronous model), then it clearly also satisfies safety under partial synchrony with up to f adversary replicas, *i.e.*, it is f-final. Perhaps more surprisingly, Theorem 4 below proves that if a protocol provides $(f+1)$-accountable safety in a delay-free network (which is the weakest form of accountable safety, since the adversary's capabilities are most constrained in the delay-free network model), it must still be the case that the protocol is f-final. This immediately implies that for many, perhaps all, network delay models, $(f+1)$-accountable safety of a protocol implies f-finality for that protocol.

▶ **Theorem 4.** *If an SMR consensus protocol provides $(f+1)$-accountable safety in a delay-free network, then it also satisfies f-finality.*

Intuition To illustrate the core idea behind the proof, we consider the scenario with $n=4, f=1$, and argue the equivalent claim that without 1-finality, there is no 2-accountable safety. For contradiction, consider a protocol executed by four replicas P_i, $i \in [4]$, that is not 1-final, yet 2-accountable safe under a delay-free network.

Consider the following executions: In world 0 (Figure 1), the network is partially synchronous, and P_4 (red in Figure 1) is adversary. Besides protocol deviations of P_4, the adversary delays messages among honest replicas to cause a safety violation (which is possible because the protocol is assumed not 1-final). In worlds 1, 2, and 3 (Figure 1), the network is delay-free, and there are three adversary replicas. The replica P_4 is adversary, and in all of these worlds behaves the same as in world 0. The remaining adversary replicas (orange in Figure 1) behave like their honest counterparts in world 0, except they *emulate* the message delivery schedule an honest replica would have had in their place in world 0, by delaying the sending/receiving of messages to/from the honest replica. Clients observing the protocol cannot distinguish between any of the worlds 0, 1, 2, and 3. Therefore, there is a safety violation in worlds 1, 2, and 3 as well.

Finally, since the SMR protocol is assumed to be 2-accountable safe in a delay-free network, the forensic protocol called by the clients identifies 2 replicas as protocol violators in each of the worlds 1, 2 and 3. However, as these worlds are indistinguishable, there is a non-negligible probability that the forensic protocol wrongly identifies an honest replica, which is a contradiction, thereby proving the original claim.

We next present the formal proof.

Proof of Theorem 4. We prove the contrapositive of the statement. For contradiction, suppose a protocol does not satisfy f-finality, yet provides $(f+1)$-accountable safety under a delay-free network. We consider a world 0 with a partially-synchronous network, and $n-f$ worlds indexed by $i \in [n-f]$ with delay-free networks.

World 0: Consider two clients cl, cl' and n replicas. The network is partially-synchronous and the adversary $\mathcal{A}_{\mathrm{p}}$ controls f replicas, denoted by $P_{n-f+1}, \ldots, P_n$. The remaining replicas are honest. Safety is violated and two (potentially the same) clients cl and cl' output conflicting ledgers with some non-negligible probability.

World i: The adversary $\mathcal{A}_{\mathrm{i}}^{(i)}$ of world i corrupts all replicas except P_i. The f replicas that were adversary in world 0 behave the same as in world 0. The remaining adversary replicas behave like the corresponding honest replicas in world 0, except they also emulate the network delay of world 0: For each message sent by P_i, adversary replicas pretend as if the message was delivered at the time slot in which it was delivered in world 0, even though it was in fact delivered instantly in world i, which is delay-free. Adversary replicas also send the same messages to P_i as in world 0, but they ensure that these messages are delivered to P_i at the same time slots within world i as they were delivered in world 0, by delaying the sending of the messages if necessary (after delayed sending, the delay-free network will deliver them instantly).

Since P_i receives the same messages at the same time slots in world i as in world 0, it cannot distinguish the worlds 0 and i. Thus, P_i shows the same behavior in both worlds.

As the adversary replicas simulate their behavior from world 0 within world i, and P_i shows the same behavior in both worlds, cl and cl' cannot distinguish the two worlds 0 and i. Thus, they output conflicting ledgers with non-negligible probability. In this case, by the assumed $(f+1)$-accountable safety of the protocol under delay-free networks, the forensic protocol, invoked with the information received by these clients from the replicas, identifies at least $f+1$ replicas as protocol violators in world i with non-negligible probability.

Finally, since the two worlds 0 and i are indistinguishable for cl and cl' for all i, the worlds $i \in [n-f]$ are indistinguishable as well for cl and cl'. Thus, since n does not scale in λ, the forensic protocol has a non-negligible probability to falsely accuse an honest replica, which is a contradiction. ◀

4 Simplification of Earlier Results

4.1 Impossibility of Accountability Follows from Impossibility of Finality

Earlier work [27, Theorem B.1] shows that no protocol can be f_{a}-accountable safe and f_{l}-live for $2f_{\mathrm{l}} + f_{\mathrm{a}} > n$ under a synchronous or partially-synchronous network. This result follows directly from Theorem 4, combined with the safety–liveness bound under partial synchrony [11, Theorem 4] restated below by Proposition 5.

▶ **Proposition 5** (From [11]). *No protocol can satisfy f_{s}-finality, and f_{l}-liveness under a delay-free network, for $2f_{\mathrm{l}} + f_{\mathrm{s}} \geq n$.*

In other words, given f_{s}, f_{l}, $2f_{\mathrm{l}} + f_{\mathrm{s}} \geq n$, no protocol can simultaneously preserve its safety under asynchrony with f_{s} adversary replicas, and remain live with f_{l} adversary replicas, even if the network is delay-free. This is stronger than the claim that no protocol provides f_{s}-safety and f_{l}-liveness under partial synchrony, yet this stronger result directly follows from [11, Theorem 4]. Equipped with Proposition 5, we can show that no protocol can be f_{a}-accountable safe and f_{l}-live under a synchronous network for any Δ, including a delay-free network, if $2f_{\mathrm{l}} + f_{\mathrm{a}} \geq n$.

▶ **Corollary 6.** *No protocol can be f_{a}-accountable safe and f_{l}-live, for $2f_{\mathrm{l}} + f_{\mathrm{a}} > n$, under a synchronous or partially-synchronous network, for any Δ.*

Proof. Towards contradiction, suppose the protocol satisfies f_{a}-accountable safety and f_{l}-liveness, for some f_{a}, f_{l} such that $2f_{\mathrm{l}} + f_{\mathrm{a}} > n$. Then, by Theorem 4, the protocol satisfies f_{s}-finality with $f_{\mathrm{s}} = f_{\mathrm{a}} - 1$ and f_{l}-liveness under the same network, for f_{s} and f_{l}, such that $2f_{\mathrm{l}} + f_{\mathrm{s}} \geq n$. This is a contradiction with Proposition 5. ◀

4.2 Availability–Accountability Dilemma Follows from Availability–Finality Dilemma

To state the availability–accountability dilemma, we recall the formal model for dynamic participation (*i.e.*, temporary crash faults) from [23], which itself was based on the *sleepy* model of consensus [24]. There exists a *global awake time* GAT, such that for every honest replica and time slot prior to GAT, the adversary $\mathcal{A}$ has the power to determine whether the replica is *awake* (*i.e.*, online) or *asleep* (*i.e.*, offline) in that slot. All honest replicas are awake after GAT. Awake replicas follow the protocol. Asleep replicas are in a state of *temporary crash fault*, and do not execute the protocol in the respective time slot. Adversary replicas are always awake. Messages sent to a replica while asleep are buffered and input to the replica whenever it turns awake again. The adversary-environment tuple $(\mathcal{A}_{\mathrm{pda}}, \mathcal{Z}_{\mathrm{pda}})$ models a partially-synchronous network with $\mathsf{GST} < \infty$ and $\mathsf{GAT} \in [0, \infty)$ that can be adaptively chosen by $\mathcal{A}_{\mathrm{pda}}$ and are not known by honest replicas or the protocol designer.

Given this model, we can prove the availability–accountability dilemma of [23] as a corollary of Theorem 4 and the blockchain CAP theorem ([16, Theorem 4.1]) restated below by Proposition 7. Let β denote the largest *fraction* across all time slots, of adversary replicas among awake replicas. In the statement below, β_{s}-safety, β_{l}-liveness, β_{f}-finality, and β_{a}-accountable safety are defined analogously to their definitions in Sections 2 and 3.

▶ **Proposition 7** (From [16]). *No protocol provides both β_{f}-finality and β_{l}-liveness for any $\beta_{\mathrm{a}}, \beta_{\mathrm{l}} > 0$ under $(\mathcal{A}_{\mathrm{pda}}, \mathcal{Z}_{\mathrm{pda}})$.*

▶ **Corollary 8.** *No protocol provides both β_{a}-accountable safety and β_{l}-liveness for any $\beta_{\mathrm{a}}, \beta_{\mathrm{l}} > 0$ under $(\mathcal{A}_{\mathrm{pda}}, \mathcal{Z}_{\mathrm{pda}})$.*

Proof. Towards contradiction, suppose $\beta_{\mathrm{a}}, \beta_{\mathrm{l}} > 0$. By Theorem 4, if a protocol provides β_{a}-accountable safety for some $\beta_{\mathrm{a}} > 0$, then it satisfies β_{f}-finality for some $\beta_{\mathrm{f}} > 0$. Thus, it satisfies β_{f}-finality and β_{l}-liveness with some $\beta_{\mathrm{f}}, \beta_{\mathrm{l}} > 0$ under $(\mathcal{A}_{\mathrm{pda}}, \mathcal{Z}_{\mathrm{pda}})$. This is a contradiction with Proposition 7. ◀

5 Finality, Accountable Safety, and Security under Partial Synchrony

Finally, we elaborate on the implications and non-implications of our result for the relations among finality, accountable safety, and security under partial synchrony. Note that even though $f+1$-accountable safety implies f-finality, $f+1$-accountable safety does not imply security under partial synchrony, even for protocols with non-trivial liveness guarantees. To see this, let us construct a protocol called SyncFin that satisfies $f+1$-accountable safety (and f-safety under partial synchrony by Theorem 4), yet cannot recover liveness after GST under a partial synchronous network, even though it is $(\lfloor (n-f)/2 \rfloor)$-live under synchrony (which is the best possible liveness resilience satisfiable by any such protocol due to Corollary 8).

The SyncFin protocol consists of two components: an underlay SMR consensus protocol executed by the n replicas and secure under Δ-synchrony (*e.g.*, Sync HotStuff [1], Sync-Streamlet [6]), and an add-on 'gadget' of 'finality signatures' on the ledgers output by the underlay protocol, which works as follows: Once a block is confirmed for the first time at some height h by the underlay protocol in the view of a replica, the replica 'votes for' the respective chain by creating a *finality signature* on the block and broadcasting it to all other replicas. A replica creates at most one finality signature per height, on the first block observed to be confirmed at that height by the underlay protocol in their view. If the replica later observes a conflicting block become confirmed by the underlay protocol at the same height, it does not sign the conflicting block (or any descendent thereof). Clients *finalize* (*i.e.*, confirm) a block of this new protocol that is a composite of underlay and finality-signature gadget, upon observing $\lfloor (n+f)/2 \rfloor + 1$ finality signatures on a block and its prefix.

▶ **Theorem 9.** *SyncFin satisfies* $(f+1)$*-accountable safety, and* $(\lfloor (n-f)/2 \rfloor)$*-liveness under synchrony.*

Proof. If two clients output two conflicting ledgers, then there must be a height, where two conflicting blocks each gathered $\lfloor (n+f)/2 \rfloor + 1$ finality signatures. Then, at least $f+1$ replicas must have created finality signatures for two blocks at the same height. Since this is a protocol violation, these replicas are adversary. Thus, in the event of a safety violation, the forensic protocol can provably identify $f+1$ adversary replicas as protocol violators by identifying these $f+1$ double-signing replicas. The forensic protocol never identifies an honest replica since an honest replica creates at most one finality signature per height.

Suppose there are $\lfloor (n+f)/2 \rfloor + 1$ or more honest replicas and the network is synchronous. Since the underlay protocol (*e.g.*, Sync HotStuff, or Sync-Streamlet) is $(\lfloor n/2 \rfloor)$-live under synchrony, a transaction input to an honest replica appears in the chain output by the underlay protocol in the view of all honest replicas within some T_{confirm} time slots under synchrony. Upon observing a new block confirmed by the underlay protocol at a certain height, each replica sends a finality signature on the block. By the $(\lfloor n/2 \rfloor)$-safety of these underlay protocols under synchrony, all replicas observe the same block at a given height if they observe a block at all. Thus, within T_{confirm} time slots of tx being input to an honest replica, all honest replicas create finality signatures for the confirmed underlay blocks containing tx and those in its prefix. Consequently, within $T_{\mathrm{confirm}} + \Delta$ time, all clients observe $\lfloor (n+f)/2 \rfloor + 1$ finality signatures on a block containing tx and its prefix blocks,

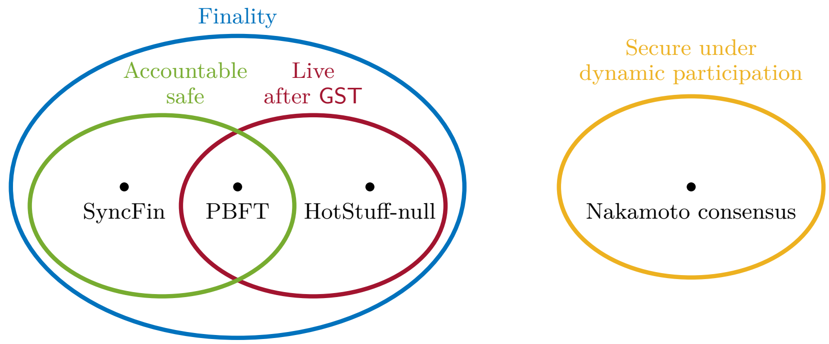

Figure 2 Venn diagram of protocols satisfying finality, accountable safety, security under partial synchrony, and dynamic participation. The key Theorem 4 of this work means that accountable safe protocols are contained in the set of final protocols.

implying that tx enters the ledgers of all clients within $T_{\text{confirm}} + \Delta$ time. This shows the $(\lfloor (n-f)/2 \rfloor)$-liveness of the protocol under synchrony. ◀

However, observe that the SyncFin protocol does not satisfy liveness under partial synchrony with any resilience greater than 1 replica. Before GST, the adversary can partition the honest replicas into two sets of equal size that cannot communicate with each other. Then, with the help of the single adversary replica, each set of honest replicas confirms a conflicting, different block, in the underlay, and sends finality signatures on their respective conflicting blocks. After signing their blocks, honest replicas would refuse to sign any conflicting block, implying that even after GST, no block ever receives finality signatures from $\lfloor (n+f)/2 \rfloor + 1$ replicas. This implies that liveness cannot be recovered after GST. SyncFin is thus an example for a protocol that is accountable safe, and has finality, but is not secure in partially-synchronous networks, because it is not live after GST.

We summarize the relationships between finality, accountable safety, security under partial synchrony and dynamic participation in Figure 2. The blue set contains protocols with $n = 3f+1$ replicas that provide f-finality and f-liveness under synchrony. The green set contains accountable safe protocols with resilience $f+1$, whereas the red one contains protocols that are f-secure under partial synchrony, *i.e.*, are f-final and f-live after GST. Since $f+1$-accountable safety implies f-finality, the green set is within the blue one. By Proposition 7, no protocol provides β_{f}-finality and β_{l}-liveness under a dynamically available network for any $\beta_{\mathrm{f}}, \beta_{\mathrm{l}} > 0$, *i.e.*, the blue and the yellow sets do not intersect (and as a consequence, the green and yellow sets do not intersect—the availability–accountability dilemma). Finally, Theorem 9 shows that SyncFin is f-accountable safe and f-live under synchrony, but as we have seen above, it is not f-live after GST under partial synchrony, so it is not in the red set. PBFT [5] is both f-accountable safe [27] and f-safe and live under partial synchrony. An example of a protocol that is not accountable safe, yet secure under partial synchrony is HotStuff-null described in [27].

References

1 Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin. Sync hotstuff: Simple and practical synchronous state machine replication. In *IEEE Symposium on Security and Privacy*, pages 106–118. IEEE, 2020.

2 Ethan Buchman, Rachid Guerraoui, Jovan Komatovic, Zarko Milosevic, Dragos-Adrian Seredinschi, and Josef Widder. Revisiting tendermint: Design tradeoffs, accountability, and practical use. In *DSN (Supplements)*, pages 11–14. IEEE, 2022.

3 Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget, 2017. `arXiv:1710.09437v4`.

4 Vitalik Buterin, Diego Hernandez, Thor Kamphefner, Khiem Pham, Zhi Qiao, Danny Ryan, Juhyeok Sin, Ying Wang, and Yan X Zhang. Combining ghost and casper, 2020. `arXiv:2003.03052v3`.

5 Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *OSDI*, pages 173–186. USENIX Association, 1999.

6 Benjamin Y. Chan and Elaine Shi. Streamlet: Textbook streamlined blockchains. In *AFT*, pages 1–11. ACM, 2020.

7 Pierre Civit, Seth Gilbert, and Vincent Gramoli. Polygraph: Accountable byzantine agreement. In *ICDCS*, pages 403–413. IEEE, 2021.

8 Thomas Dinsdale-Young, Bernardo Magri, Christian Matt, Jesper Buus Nielsen, and Daniel Tschudi. Afgjort: A partially synchronous finality layer for blockchains. In *SCN*, volume 12238 of *LNCS*, pages 24–44. Springer, 2020.

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

10 Xinshu Dong, Robin Linus, Orfeas Stefanos Thyfronitis Litos, Ertem Nusret Tas, David Tse, and Fisher Yu. Bitcoin staking: Unlocking 21m Bitcoins to secure the proof-of-stake economy. URL: `https://docs.babylonchain.io/assets/files/btc_staking_litepaper-32bfea0c243773f0bfac63e148387aef.pdf`.

11 Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.

12 Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

13 Seth Gilbert and Nancy A. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

14 Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. Peerreview: practical accountability for distributed systems. In *SOSP*, pages 175–188. ACM, 2007.

15 Andreas Haeberlen and Petr Kuznetsov. The fault detection problem. In *OPODIS*, volume 5923 of *LNCS*, pages 99–114. Springer, 2009.

16 Andrew Lewis-Pye and Tim Roughgarden. How does blockchain security dictate blockchain implementation? In *CCS*, pages 1006–1019. ACM, 2021.

17 Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

18 Ryuya Nakamura. Hierarchical finality gadget. URL: `https://ethresear.ch/t/hierarchical-finality-gadget/6829`.

19 Joachim Neu, Srivatsan Sridhar, Lei Yang, and David Tse. Optimal flexible consensus and its application to ethereum. Cryptology ePrint Archive, Paper 2023/1211, 2023. URL: `https://eprint.iacr.org/2023/1211`.

20 Joachim Neu, Ertem Nusret Tas, and David Tse. Snap-and-chat protocols: System aspects, 2020. `arXiv:2010.10447v1`.

21 Joachim Neu, Ertem Nusret Tas, and David Tse. The availability-accountability dilemma and its resolution via accountability gadgets, 2021. `arXiv:2105.06075v3`.

22 Joachim Neu, Ertem Nusret Tas, and David Tse. Ebb-and-flow protocols: A resolution of the availability-finality dilemma. In *IEEE Symposium on Security and Privacy*, pages 446–465. IEEE, 2021.

23 Joachim Neu, Ertem Nusret Tas, and David Tse. The availability-accountability dilemma and its resolution via accountability gadgets. In *Financial Cryptography*, volume 13411 of *LNCS*, pages 541–559. Springer, 2022.

24 Rafael Pass and Elaine Shi. The sleepy model of consensus. In *ASIACRYPT (2)*, volume 10625 of *LNCS*, pages 380–409. Springer, 2017.

25 Alejandro Ranchal-Pedrosa and Vincent Gramoli. Zlb: A blockchain to tolerate colluding majorities, 2020. `arXiv:2007.10541v3`.

26 Suryanarayana Sankagiri, Xuechao Wang, Sreeram Kannan, and Pramod Viswanath. Blockchain CAP theorem allows user-dependent adaptivity and finality. In *Financial Cryptography (2)*, volume 12675 of *LNCS*, pages 84–103. Springer, 2021.

27 Peiyao Sheng, Gerui Wang, Kartik Nayak, Sreeram Kannan, and Pramod Viswanath. BFT protocol forensics. In *CCS*, pages 1722–1743. ACM, 2021.

28 Alex Skidanov. Fast finality and resilience to long range attacks with proof of space-time and Casper-like finality gadget, 2019. Whitepaper. URL: `http://near.ai/post`.

29 Alistair Stewart and Eleftherios Kokoris-Kogia. Grandpa: a byzantine finality gadget, 2020. `arXiv:2007.01560v1`.