Malicious Security Comes for Free in Consensus with Leaders

Matthieu Rambaud*

Telecom Paris, Institut polytechnique de Paris, France

Abstract. We remove the so far quadratic bit communication cost of three desirable properties of consensus protocols with leaders: *Responsiveness with Optimal latency*, *Optimistic Fast Track* and *Strong Unanimity*. No existing consensus protocol with leaders with subquadratic bit complexity has any of those properties so far. [Hotstuff, Podc'19] has suboptimal latency of two more messages delays, whereas Hotstuffv1 is not responsive. [SBFT, Dsn'19] has quadratic complexity every time a new leader appears. Strong Unanimity has equally quadratic complexity so far [Chan et al Podc'19] in this setting. We reduce the communication costs of each of these properties down to $O(n \log n)$. In addition, we achieve them *simultaneously*, with optimal corruption threshold.

To achieve these specifications we use the structure of the consensus of Castro-Liskov / [SBFT, Dsn'19], in which we use succinct (range-) proofs of knowledge as a drop-in replacement for the forwarding of many messages. We use the same kind of strategy to enable a Fast Track and Strong Unanimity. Namely, we incorporate the additional structure of [SBFT, Dsn'19] and of [Chan et al Podc'19] in the previous protocol. Which we instantiate with proofs of knowledge of: a set of signed messages, from a threshold number of issuers, in which no value appears in majority. The required proofs of knowledge can be obtained from any succinct proof system. Of independent interest, we also introduce alternative elementary proofs, solely based on a black box Threshold Signature Scheme (TSS).

Applied to the state of the art leader-less fully asynchronous consensus protocol [Podc'19], which uses the [Hotstuff, Podc'19] consensus as baseline, this reduces its latency by 25%. This speedup directly carries over the state machine replication system [Hotstuff, Podc'19], and thus to Libra. Of independent interest we maintain linear complexity when requiring both External Validity and Halting in finite time, in the Amortized regime over long values. Instantiated with the recent unpublished logarithmic Transparent TSS of Attema et al, none of our protocols requires a trusted setup or a distributed key generation.

1 Introduction

In §1.1 we identify three complexity limitations in the state of art leader-based consensus protocols [AGM18, YMR$^+$19, GAG$^+$19, ACD$^+$20]. In §1.2 we present the technical roadblocks to removing these limitations. As [GAG$^+$19], we use as guiding line a variation on the older [CL99] protocol, denoted "PBFT", to illustrate our point. The reason is that, surprisingly, it is on [CL99] that our approach applies naturally. In §1.3 we state our contributions, and sketch the technical hurdles that remain nontrivial at this point of the exposition.

1.1 Current Limitations in the Complexity of Consensus with Leaders

Let us now consider consensus protocols for $n = 3t + 1$ players in an asynchronous communication network. t are maliciously corrupted, the others are denoted as "honest".

Definition 1. Let Π be a protocol in which all n players supply an input value and outputs at most one output value. Then, Π is called a *consensus* protocol if it satisfies the following properties:

– **Consistency:** No two honest players output different values;
– **Consensus Weak Unanimity (CWU):** *If all* n players are honest *and* have the same input, then this is the only value that they can possibly output [Ben83];

*rambaud@enst.fr

Those conditions are satisfied by the trivial protocol where players do nothing and never output. Additional requirements that players output in finite time are called *Liveness* conditions. We will review some, then, in Definition 4 we state the one that will be considered in our illustration.

CWU has the inherent limitation that, despite all honest players having the same input value, they may still output another value. Anticipating, this will be clear from Figure 1 **1**. This is why in the literature, "consensus" is instead mainly understood as requiring the:

Definition 2 (CSU)**.** A consensus has *Strong Unanimity*, if as soon as all *honest* players have the same input value, then this is the only possible output.

Next, a much investigated additional feature in consensus ([Lam02, Zie06, BIW10, PS18, KAD$^+$09, ANRS20, MCK20, Ram20]) is to enable players to hope for the best and `output` sooner if possible. We will use the canonical measurement [Lam06, §2.3] of time in a message-passing network. Namely, we say that some event, in some execution of a protocol, happens within *d messages delay*, if d is the size of the longest chain of consecutive messages sent-then-received until this event happens.

Definition 3 (Fast Track)**.** We say that a consensus has an (optimistic) Fast Track if we have furthermore that: if all players are honest and have the same input, then all players `output` within two messages delay from the start of the execution.

Under partial synchrony where messages can be lost (§3.3, [DLS88, §3.1]), this holds only if the network is initially synchronous. Liveness is conditioned to the following abstract entity, denoted as "Pacemaker" in [YMR$^+$19]. It maintains a monotonically increasing *phase counter* ϕ (the terminology of [CS06, §3.1]), along with the identity of a specific player L_ϕ per phase. This player L_ϕ is denoted as the *Leader* of phase ϕ. The Pacemaker sends these informations to players by messages on the network. We say that a player "is in phase ϕ" if ϕ is the highest phase number that he was notified of so far. Then, output is guaranteed if: the leader L_ϕ of some phase ϕ is honest, and if all honest players are simultaneously in this phase for *long enough.* Or said otherwise, and if the network is *fast enough* while all players are in ϕ. This is captured by the following Definition (we recall that messages are assumed to be delivered in finite time).

Definition 4 (Optimal Latency)**.** Under this model, we say that a consensus in the sense of Definition 1 has *Optimal Latency* if: If the Pacemaker remains blocked on the same phase number ϕ forever, and such that the leader of this phase is honest, then, every player is guaranteed to `output` within 6 consecutive messages exchanged between himself and the leader, from the point where he is in this phase.

If the consensus has neither Fast Track nor Strong Unanimity, *then in addition*, the latency is required to be of 5 messages delays in the first phase $\phi = 1$ specifically (still 6 in higher phases).

The first limitation of the state of the art, is that [YMR$^+$19] has not optimal latency (nor Fast Path nor Strong Unanimity). Indeed each phase takes 8 messages delay instead of 6). By contrast, [AGM18, v1] suffers from another limitation, pointed in [YMR$^+$19], which is that it is *Not Responsive* in the sense below. That is, the leader of a new phase is instructed to *wait* for some arbitrary *fixed delay* Δ before he can send new messages. This waiting instruction holds irrespective of the current network delays. It holds in particular even in some favorable execution in which all messages happen to be actually delivered much faster than Δ (including those of the Pacemaker).

Definition 5 (Responsiveness)**.** We say that a distributed protocol is Responsive if players have no instruction to wait for some fixed delay, at any point in the protocol.

Then we will see in §1.2.4 why the protocol of [GAG$^+$19], denoted "SBFT", which has an alternative form of Fast Track (see Definition 1 and construction in §8.2.3), has *quadratic* $\Omega(n^2)$ bit complexity of communications in phases higher than $\phi = 1$. Finally the only known consensus (in this model) known to us with Strong Unanimity, which is [ACD$^+$20, §6.1 long version], is Non Responsive and has $\Omega(n^2)$ communication complexity. We now study in more details the roadblocks to removing all the limitations mentionned.

Definition 6. The *bit complexity* in a phase is the worst case total number of bits *sent by all honest players.* The *bit complexity of a protocol* is the *worst case* bit complexity in a phase, over all phases over all executions.

1.2 Roadblocks to Subquadratic Bit Complexity

1.2.1 Further context on the Definitions of §1.1

About Definition 1. Consistency is implicit, e.g., in the definition of blockchain ([BMTZ17]), since all players agree on an old enough prefix. CWU is the only guarantee of most protocols [CL99, CM19, DGKR18, YMR+19, AMN+20, CS20] in which ordered sequences of consensus are considered. This setting is historically called "BFT" or "state machine replication".

About Definition 4. It is proven in [FLP85] that no consensus protocol can guarantee that all players output in finite time in every execution (excepted protocols tolerating no corruption). Thus various Liveness conditions circumventing this impossibility are considered in the literature. One line of work consists in tolerating a probability of failure, which is exponentially small in the latency and/or communication complexity. However, all such probabilistic protocols achieving quasi linear communication complexity, require suboptimal adversary bound (see §1.4). The Liveness condition that we consider in this work instead was initiated concurrently in 1988 by [DLS88, CL99, Lam98]. Their protocol guarantee `output` in any execution in which, at some point, the following conditions are matched:

(1) the identity of some specific player, denoted as being the "Leader", is made known to all honest players by some mechanism;

(2) after this designation, the network is fast enough.

Or, equivalently, honest players are simultaneously in the same phase with a honest leader, for a time which is long enough compared to the network delays. Actually, since the network is asynchronous, the notion of "fast" or "simultaneous" has actually no sense. This is why Definition 4 captures this instead by assuming that the favorable phase number holds forever. The notion of time before `output` being then captured by the maximum number of consecutive messages in this favorable phase. In practical mechanisms, if messages are delivered too slowly, then the leader is assumed faulty and replaced.

Interest of such protocols with the "honest& timely leader" liveness condition of Definition 4 are multiple:

- they are massively used as such ([Lam98, Lib19, CL99]);
- they are used as a *building block* in the best-known fully asynchronous probabilistic consensus ([AMS19], with a quadratic complexity);
- they are finally used as a *baseline* in committee-based protocols, such as [ACD+19].

In the crash-fault tolerant consensus [Lam98], every player can start to play the role of the leader at any point. Output is then guaranteed in executions in which exactly *one* player plays this role from some point. But as noticed in [Lam11, §8], this condition has an interest only provided additional mechanisms to enforce that not multiple malicious players designate themselves as leader repeatedly. Therefore, leader designation and leader replacement are abstracted out by a global *Synchronizer* entity [NBMS19, NK20, BCG20b]. In Definition 4 we consider for simplicity the one of [YMR+19], called "Pacemaker".

Optimality in Definition 4 Optimality of the delay of 6, for protocols in our model that have *subquadratic* complexity is proven in [Ram20, C']. In our same model, optimality of the delay of 5 in the first phase is proven in [Ram20, A'], both for the class of protocols with linear message complexity. In particular, [AGM18, v1] *has* Optimal Latency (6 messages delay, and even 5 in the first phase since it is neither Fast Path nor Strong Unanimity). Notice that the alternative Fast Track (Definition 1 and construction §8.2.3) of [GAG+19], in 3 messages delay, circumvents [Ram20, B] and thus [GAG+19] still achieves also 5 messages in the first phase.

Although not considered in this paper, is one stands for protocols with higher message complexity, then the optimal is brought down to 4 ([Ram20, C]), and 3 in the first phase (by [Ram20, A], when no fast Track nor CSU), as achieved in [CL99, DGV05, AGK+15] thanks to all-to-all messages.

About Definition 3. fast track is even possible in 1 message delay, but then this comes at the cost of for all-to-all communication [Zie06, Kur02] (at least if handling for maximal adversary bound $n = 3t + 1$). Being concerned with linear communication complexity, we thus consider Fast Tracks in 2 messages delay in Definition 3. See Definition 1 and construction §8.2.3 for a variation in 3 messages.

About Definition 5. For instance, [ACD$^+$20, §6.1 long version] instructs players to wait for an estimate Δ of the network delay before sending new messages. Even if this estimate is updated on the fly in loc. cit., this protocol is thus anyway Not Responsive. Nonetheless, any nontrivial distributed message-passing protocol *does* instruct players to wait to receive some *messages* before they can send new ones. E.g. when players have finished all instructions that they could perform in a phase, they wait de facto for a message from the Pacemaker. To be sure, the synchronizers of [NBMS19, NK20, BCG20b] are implemented with waiting instructions. Using such a Synchronizer model (our "Pacemaker") enables to abstract out the waiting instructions in its implementation. Definition 5 being stated in the Pacemaker model, this thus make our point clearer, e.g., by emphasizing that [AGM18] is Not Responsive even under this abstraction.

About Definitions 6 and 4, stated as a worst case for honest players. An arbitrary number of phases can happen before the conditions of Definition 4, that guarantee output, are matched (honest leader and a network fast enough). Likewise for latency, recall that this measurement counts the number of consecutive messages delivery delays of type leader-to-players then players-to-leader. Thus, we cannot prevent one phase from taking a long time and a dishonest leader interacting one-by-one with every honest players consecutively. This is why, instead, Definition 4 defines Optimal Latency only in phases with a honest leader. Let us also remark nevertheless that the latencies that we claim in our protocols do actually also bound the maximal number of consecutive interactions of a given dishonest leader and any given fixed honest player.

1.2.2 The So-far Quadratic Cost of *Responsiveness with Optimal latency*

In short, the current situation is that the only known *responsive* consensus with worst-case bit complexity of communications in $O(n)$, which is [YMR$^+$19], has suboptimal latency. As a matter of fact, it is explained in introduction of [YMR$^+$19] that they opted for *two more* consecutive round-trips of messages between the leader and players, compared to the optimal [AGM18], to achieve responsiveness with linear communication complexity. In what follows we are going to detail the hurdles for achieving simultaneously Responsiveness with Optimal Latency, while preserving quasi linear communication complexity.

Illustration with the Baseline [CL99, PBFT] Let us illustrate the problem of bit complexity with the baseline Consensus with Weak Unanimity of [CL99], which is denoted as "PBFT". In each phase, players follow the same set of instructions. The first phase $\phi = 1$ distinguishes itself in that the instructions are simpler than in higher phases $\phi \geq 2$. So we start by describing $\phi = 1$. To make our point clear, we modify PBFT by replacing all-to-all messages, by, instead communication patterns of type leader-to-all and all-to-leader. Indeed otherwise achieving subquadratic complexity would be anyway hopeless. We describe in Figure 1 the actions that players are instructed to do when they are in $\phi = 1$, of which we note $L_{\phi=1}$ the leader. The protocol is *responsive* (Definition 5): players in phase $\phi = 1$ perform these actions *as soon as they can.* The numbered steps *do not* denote any waiting instruction. The three message types that need to be made explicit in the specification of the protocol below, are denoted as: propose, lock vote, decision vote. Recall [ACR20, §5] that for any integer k, a Threshold Signature Scheme (TSS) provides an algorithm k-AGGREGATE that, on input a set $\mathcal{S}$ of k messages of identical content m, signed by any k out of n players, outputs a proof of knowledge of such a set $\mathcal{S}$. Unless specified differently, we will always consider a threshold $k = 2t + 1$. For clarity, we denote as lock certificate the data type output by AGGREGATE on messages of type lock vote, and decision certificate the data type output by AGGREGATE on messages of type decision vote

Overall complexity of Figure 4 Honest players send a total number of bits equal to $O(n)$ times the size of a threshold signature. The latter can be implemented constant size assuming a trusted setup ([Sho00, TCZ$^+$20]), or, $\log(n)$ size without ([ACR20]). Thus we have overall (quasi) linear complexity. This variation on [CL99] that we are considering, with communication pattern "star-shaped" around the leader, has the same latency of 5 in the first phase as [AGM18]. To be sure, the same star-shaped variation of PBFT [CL99] is also partially considered in [GAG$^+$19] under the name "linear PBFT". But they describe *only the first phase*, although we are going to see now that *the complexity issues appear instead in higher phases* $\phi \geq 2$. Informally, PBFT enforces that every player casting a "lock vote(v, ϕ)", must be "convinced" beforehand

1 Propose Leader L_ϕ multicasts $\mathsf{propose}(v_L, \phi)$ where v_L is his input.

2 Lock vote Every player, upon receiving $\mathsf{propose}(v, \phi)$ from L_ϕ for the first time (whatever v), replies with a signed "$\mathsf{lock\,vote}(v, \phi)$".

3 Lock certificate Leader L_ϕ, upon receiving $\mathsf{lock\,vote}$s for the same (v, ϕ) from $2t + 1$ distinct players, AGGREGATE them into a "$\mathsf{lock\,certificate}(v, \phi)$", which he multicasts.

4 Decision vote Every player, upon receiving from L_ϕ a $\mathsf{lock\,certificate}(v, \phi)$ for the first time (whatever v), replies with a signed $\mathsf{decision\,vote}(v, \phi)$.

5 Decision certificate Leader L_ϕ, upon receiving $\mathsf{decision\,vote}(v, \phi)$ from $2t + 1$ distinct players for the same v, AGGREGATE into a $\mathsf{decision\,certificate}(v)$, which he multicasts.

6 Output Upon receiving a $\mathsf{decision\,certificate}$ for v, a player $\mathsf{outputs}$ v.

Fig. 1. PBFT leader-centered: first phase $\phi = 1$, with linear bit complexity

that no $\mathsf{decision\,certificate}(v_j)$ for any conflicting $v_j \neq v$, could be formed relatively to a lower phase $\phi_j < \phi$ (and thus possibly output by some isolated player). The first step towards this "convincing" is that we have the invariant that, as soon as some value v_j is output in some phase ϕ_j, then there exists at least $t+1$ honest players who "saw" a $\mathsf{lock\,certificate}(v_j, \phi_j)$, and thus at least one of them will be present in any set of $2t + 1$ players. But still, after many phases, many values v_j can possibly have been in $\mathsf{lock\,certificate}$s. The simple but strict rule chosen by [AGM18],[ACD$^+$20, §6.1 long version] (and [ADD$^+$19, AMN$^+$20]) is that players $\mathsf{lock\,vote}(v, \phi)$ only if the leader exhibits to them a $\mathsf{lock\,certificate}(v, \phi_{max})$, such that they did not see any $\mathsf{lock\,certificate}(v_j, \phi_j)$ for a conflicting value v_j in a higher phase $\phi_{max} < \phi_j$. The drawback is that, mechanically, output is then guaranteed in ϕ *only* if the leader collected *all* the highest seen $\mathsf{lock\,certificate}(v_j, \phi_j)$ of *all* honest players: see the discussion in [YMR$^+$19] ("Livelessness with two-phases"). This is why the leader must *wait* for the eventual upper bound on the network delay: Δ, before processing his mailbox. Notice that [YMR$^+$19] has a related rule, while guaranteeing responsiveness, but at the cost of two extra messages delay. The rule of PBFT [CL99] is more permissive, since players have the right to $\mathsf{lock\,vote}$ even if they saw a higher $\mathsf{lock\,certificate}$ for a conflicting value. This is the key to Responsiveness with Optimal Latency. The counterpart is that the leader must append more justifications to "convince" players to $\mathsf{lock\,vote}$ a value, which makes the bit complexity explode. Namely, the previous instructions for $\phi \geq 2$ are modified as described in Figure 2: a preliminary Report step is added, then the leader appends extra justifications to his messages in **1**, then players check them in **2**.

0 Report Every player P_i sets $\phi_i \leq \phi$ the highest phase up to ϕ for which he saw a $\mathsf{lock\,certificate}$, or $\phi_i = 0$ if he did not see any so far. He sends to the leader L_ϕ: a $\mathsf{report}(\phi_i, \phi)$, appended with a $\mathsf{lock\,certificate}(v_i, \phi_i)$ if $\phi_i \geq 1$.

1 Propose The leader L_ϕ, upon receiving a set $\mathcal{R}$ of report messages from $2t + 1$ distinct players, then, letting ϕ_{max} be the highest ϕ_j in $\mathcal{R}$:

(i) either $0 < \phi_{max}$, thus he received at least one $\mathsf{lc}_{max} := \mathsf{lock\,certificate}(v_{max}, \phi_{max})$. Then he multicasts $\mathsf{propose}(v_{max}, \phi)$, appended with the justification $\{\mathcal{R},\ \mathsf{lc}_{max}\}$;

(ii) or, he sets v_L equal to his own input and multicasts $\mathsf{propose}\big(v_L, \phi)$ appended with the justification $\mathcal{R}$.

2 Lock vote Every player, upon receiving $\mathsf{propose}(v, \phi)$ from L_ϕ for the first time (whatever v), appended with a set $\mathcal{R}$ of report messages relative to ϕ, checks if:

(i) *Either* the justification comes with some $\mathsf{lock\,certificate}(v, \phi')$ for v, such that ϕ' is the highest ϕ_j reported in $\mathcal{R}$;

(ii) *Or* the justification contains no $\mathsf{lock\,certificate}$, and *all* messages in $\mathcal{R}$ report $\phi_j = 0$;

then if the check passes, he replies with a signed $\mathsf{lock\,vote}(v, \phi)$.

Fig. 2. PBFT leader-centered: higher phases $\phi \geq 2$, with *quadratic* bit complexity in **1**

Since $\mathcal{R}$ has size $\Omega(n)$, then every propose message has bitsize $\Omega(n)$ and thus the whole has *quadratic* bit complexity. Although, to be sure, we followed the optimisation (credited to [CL02] in the v2 18 Oct 2018 of [YMR+19], under the name "Strawberry") where the leader *does not* append every reported lock certificate(v_j, ϕ_j) to the set $\mathcal{R}$ that he forwards to players. Consider any phase satisfying the requirements of Definition 4. More generally, any phase with a honest leader and such that the network allows for 6 consecutive messages to be delivered before the Pacemaker issues a new phase. Then all honest players output in this phase. The intuition of the proof of safety of the consensus of Figure 2 follows from the three following invariants. We will give a formal proof of our improved protocol, that has the same formal structure, in Lemma 10 and Proposition 9.

- A set $\mathcal{R}$ of report messages from $2t+1$ distinct players who all claim: to be in phase ϕ, and to have seen no lock certificate in a higher phase than $\phi_{max} \leq \phi$; is a proof that strictly less than $t+1$ honest players did see a lock certificate in a higher phase than $\phi_{max} \leq \phi$ while being in a phase up to ϕ.
- In turn, this proves that no decision certificate is formed relatively to any higher phase ϕ' up to ϕ: $\phi_{max} < \phi' \leq \phi$, in the whole execution.
- Thus if a lock certificate is formed in ϕ for some value, then, no decision certificate is formed for any conflicting value between ϕ_{max} and ϕ, in the whole execution. And thus no conflicting honest output is triggered between those two phases. We repeat the argument with $\phi := \phi_{max}$ and conclude by backward induction.

1.2.3 The So-Far Quadratic Cost of Strong Unanimity (CSU)

In the model considered, the only CSU known to us is the [ACD+20, §6.1 long version]. But it is constructed on the top of [AGM18], so with the aforementionned stricter rule, which is incompatible with Responsiveness with Optimal Latency. So we will instead use as baseline the variant of PBFT described in §1.2.2. The modifications to this baseline to achieve CSU are described in Figure 3. First, there is a Report step even when $\phi = 1$, then an additional justification in **1**, then additional check in **2**.

0 Report A player P_i, in addition to the report message, sends to the leader L_ϕ a signed declare(v_i) where v_i is his input.

1 Propose The leader L_ϕ, upon receiving a set $\mathcal{R}$ of report messages from $2t+1$ distinct players, then, letting ϕ_{max} be the highest ϕ_j in $\mathcal{R}$:

(i) either $0 < \phi_{max}$, then as in Figure 2 **1**(i)

(ii) or: let $\mathcal{D}$ be the set of $2t+1$ declare messages received. The leader selects any value v_L reported in $\mathcal{D}$, such that *no conflicting value* $v' \neq v_L$ is reported identically in $t+1$ messages of $\mathcal{D}$. Then he multicasts propose$\big(v_L, \phi)$ appended with the justification $\mathcal{D}$ (in addition to the justification $\mathcal{R}$).

2 Lock vote Is enriched in case (ii): in addition to the previous check on $\mathcal{R}$, every player also checks that *no other value* $v' \neq v$ is reported identically in $t+1$ messages or more in the set $\mathcal{D}$. Then if this checks also passes, he replies with a signed lock vote(v, ϕ).

Fig. 3. Adding Strong Unanimity: modifications for all ϕ

So now the leader also forwards to every player a set of messages, $\mathcal{D}$, which is also itself of bitsize $\Omega(n)$. *This is the quadratic communication bottleneck of Strong Unanimity*, which comes in addition to the previous one of Figures 1&2, which was due to $\mathcal{R}$. The check performed in **2** guarantees that honest players do not cast a lock vote for some value v if there exists a conflicting value $v_{un} \neq v$ which is a unanimous input of all honest players. This is formalized in Definition 11 and in the easy Lemma 25. Thus Strong Unanimity is guaranteed.

1.2.4 The So-Far Quadratic Cost of a Fast Track

Up to now, *even without requiring Responsiveness*, the task of having a Fast Track with overall worst-case subquadratic bit complexity remained an unsolved challenge. This was raized by the authors of the *nonresponsive* [AGM18][1]. In this paper we require consensus with linear communication complexity per leader phase. So for such protocols, then a fast track takes at least 2 rounds (all-to-leader then leader-to-all: see below or in Lemma 24). Whereas 1 round is achievable without this complexity restriction. Let us also point the fact that enabling a fast track in consensus, even longer than these optimals (2 resp. 1), poses challenges: see [AGM$^+$17, SK19a, SK19b] for safety or liveness violations in such published protocols.

The protocol [GAG$^+$19, DSN'19] ("SBFT"), also from authors of [AGM18], is the state of the art among the ones with a Fast Track with linear communication complexity. Its fast track is in a suboptimal 3 rounds, instead of the optimal 2. On the other hand, having this suboptimally fast track enables the authors of [GAG$^+$19] to achieve linear communication complexity *in the whole first phase* (loc. cit. §E). In short, referring to Figure 1: this enables to factorize the vote step of **2** for both the fast track and the normal track. So the two possibly voted values for both tracks are automatically the same, so no extra justification is required from the leader. This also enables them to achieve latency of 5 rounds in the first phase (as proven tight in [Ram20, Theorem A']). But the complexity problem hits in higher phases (loc. cit. §G), which have a *quadratic bit complexity of communications*. Namely, a new leader in SBFT forwards a large set of messages to all players, as in §1.2.3. In any case, their method would not enable to guarantee Strong Unanimity *and* preserve the linear complexity in their first phase.

But, building on our CSU (§1.2.3) instead above, then obtaining a fast track with a tight 2 rounds can be done simply as follows. Upon receiving messages $\mathsf{declare}(v)$ for the same v from *all* the $n = 3t + 1$ players (denoted "fast quorum" in the folklore), then the leader L_ϕ (whatever ϕ) aggregates them with a $(3t + 1)$-threshold signature into a $\mathsf{fast\,dec\,cert}(v)$ that he multicasts to players. This enriched construction brings no overhead in bit complexity nor latency. To prove safety of it, considering that our baseline construction of §1.2.3 has Strong Unanimity, we are thus brought down to proving that: no (isolated) player can output some value v from a $\mathsf{fast\,dec\,cert}(v)$, while another player outputs a conflicting v' from a (slow) $\mathsf{decision\,certificate}(v')$. But a $\mathsf{fast\,dec\,cert}(v)$ exists only if all $2t + 1$ honest players have the same input v. Thus by CSU of the baseline, no $\mathsf{decision\,certificate}(v')$ for a conflicting value can ever be formed. So we see that, having enforced the CSU property previously in §1.2.3 trivializes the problem of the linear complexity of a Fast Track. See Lemma 24 for a formalization.

To be complete, notice that enforcing Strong Unanimity brings an unavoidable *one more* message delay (6) in the first phase ([Ram20, B']). To be sure, enforcing a plain 2 messages Fast Track in the sense of Definition 3 would anyway *also* bring this extra delay ([Ram20, B']). This explains the Definition 4. However this extra delay *can* be circumvented by enforcing instead *directly an alternative* Fast Track in 3 messages (Definition 1), as described in §8.2.3 (following [GAG$^+$19]).

1.3 Contributions

1.3.1 Communication Complexity of Consensus

Main Theorem 7. *Consider $n = 3t + 1$ players of which t are malicious, in the synchronizer model. Then there exists a consensus which is* Responsive with Optimal latency *(Definitions 5 and 4), a* Fast Track *(in two messages delay),* Strong Unanimity*; and still: a worst-case $O(n \log n)$ total bits sent by all honest players per phase.*

The protocol requires no trusted setup nor distributed key generation.

Of independent interest, our *intermediary* step, which we build in §2.1 (proven in Proposition 9), is actually a consensus with standalone *Responsiveness with Optimal latency*, in the sense of Definition 4, in that it achieves even 5 messages delay in the first phase. Only then, on the top of this consensus, we furthermore add the *Fast Track* and *Strong Unanimity*, which then increments the latency in the first phase to 6, so this has still *Optimal Latency* by Definition 4.

[1] In [AGM18, v1, March 2018, §8]: "One question left open by this work is whether linear phase-change is possible for BFT protocols with *speculatively fast tracks*. In all the known methods [MA05, Kur02, KAD$^+$09, CKL$^+$09, AGK$^+$15, GV10], we can achieve a linear reduction, either by applying LVC [linear phase change], or by using threshold signature schemes (as demonstrated in [GAG$^+$19]). However, combining the two to get linear-over-linear reduction is not obvious."

Let us recall and specify the definition of Responsive with Optimal latency, as defined in Definitions 5 and 4. This means that the consensus has the guarantee that, if the leader of some phase is honest and all honest players are in this phase during long enough (equivalently: if the network is fast enough during this phase), then, they all output within the *actual delay* of 3 round trips of messages between the leader and a quorum of players. For instance, if all messages are actually delivered within δ, then the Latency is 6δ. This constrasts with the $O(\Delta)$ of [YMR+19] for some predefined Δ, possibly much larger than the actual network delay δ.

1.3.2 Sketch of the Consensus of Main Theorem 7

The detailed construction and proofs are done in §2.

Responsiveness with Optimal Latency The solution consists in replacing the set $\mathcal{R}$ forwarded in the **1** of Figure 2 of §1.2.2. In place of $\mathcal{R}$, the leader forwards a succinct *Proof of Knowledge* of a set $\mathcal{R}$ of authenticated messages from $2t+1$ players, such that, all numbers ϕ_i reported in these messages *are lower than or equal to* the public number ϕ_{max}. Thus this proof, which we call *Proof of NonSupermajority*: PnS, enables players to VERIFY that the claimed set $\mathcal{R}$ (which they do not see anymore), has indeed the property required in **2**.

We thus obtain Responsiveness with Optimal Latency. In particular, since we are using Figure 1 at this stage as a baseline for the first phase $\phi = 1$, this yields the claimed 5 rounds of Optimal Latency in the first phase (Definition 4). A PnS of size $\log(n)$ may be constructed from generic proof systems without trusted setup. Such as the ones [BBHR18] (used to verify transactions in blockchains [Tru20]), or Snarks (as used in [BCG20a], to aggregate signatures in a distributed way). So the total bitsize of the PnS sent to all players is $n\log(n)$. Then we have remaining messages, of which those sent by the leader in the two last steps contain threshold signatures. *Instantiating* with the TSS of [ACR20, Thm 4], which are of size $\log(n)$, therefore does not augment the complexity. Furthermore these TSS do not require any trusted setup no distributed key generation, which makes our claim. So this concludes our complexity claim. But for sake of illustration, we will describe anyway an elementary PnS in §6.

Adding Fast Track and Strong Unanimity The solution for Strong Unanimity (and thus Fast Track by Lemma 24) consists in replacing the set $\mathcal{D}$ forwarded in the **1** of §1.2.3 Figure 3, by a succinct *proof of knowledge* of such suitable set $\mathcal{D}$ of authenticated messages from $2t+1$ players. Namely this proof, which we call *Proof of Exclusivity*: PoE, enables players to VERIFY, as in **2** of Figure 3, that the claimed set $\mathcal{D}$ (which they do not see anymore), has indeed the property that *no other value $v' \neq v$ is identically reported $t+1$ times or more in $\mathcal{D}$*. A PoE of $\log(n)$ size may be constructed from generic proof systems, such as [BBHR18, BCG20a, AC20]. But for sake of illustration, we will describe anyway elementary protocols to generate PoE with suboptimal size in §5 (under a weakened-but-sufficient Definition 15 of a PoE).

1.3.3 Of Independent Interest: Alternative Constructions with Elementary Techniques

We consider an arbitrary Threshold Signature Scheme (TSS), instantiated twice (with threshold $t+1$ for one instantiation and $2t+1$ for the other). Let us denote by $|TSS|$ the size of a threshold signature, and V the size of the inputs range. For instance, the TSS of [ACR20, Thm 4] has $|TSS| = O(\log n)$ size. For simplicity we state the complexities below for the case of such a TSS with nonconstant size : $|TSS| > O(1)$. We thus do not specify the other linear terms in n, that are explicit from the subsequent protocols. Instantiating with optimized elementary techniques of independent interest, introduced below in §1.3.4, §1.3.5, §1.3.6, we have:

Theorem 8. *In the same model, there exists a consensus which is* Responsive with Optimal latency *(Definitions 5 and 4), a* Fast Track *(in two messages delay),* Strong Unanimity*; with a worst-case $O(n\phi + n\log V|TSS|)$ total bits sent by all honest players in a given phase ϕ.*

Removing Responsiveness, then the bit complexity per phase drops to $7n|TSS|$.

Complexity claim of $7n|TSS|$ when removing Responsiveness Anticipating on the details of the next §1.3.4, §1.3.5 and §1.3.6, the claimed $7|TSS|$ when standing for NonResponsiveness, is achieved by:

– using the NonReponsive consensus of §4.3.3 (which is the one of Hotsuffv1 [AGM18]) as baseline
– Tweak it such that it can be combined with a black box *interactive* protocol that produces a PoE in 2 round trips. This is specified in Figure 8
– Instantiate the PoE using the NonResponsive interactive protocol presented in §5.2. It has size $5|TSS|$, to which we add the $2|TSS|$ sent by the leader to each player in a phase (the certificates).

Complexity claims of $O(n\phi + n \log V|TSS|)$ when requiring also Responsiveness Anticipating on the details of the next §1.3.5 and §1.3.6, the claimed additional cost when standing for NonResponsiveness, is achieved by:

– Tweak the baseline of Figure 3 such that it can be combined with a black box *interactive* protocol that produces a PoE in 2 round trips. This is specified in Figure 8
– Instantiate the PoE using the Responsive interactive protocol presented in §5.1, which produces PoE of sizes $\log V|TSS|$.
– Instantiate the PnS using the Responsive elementary protocol presented in §6 (Proposition 22), whose construction requires players to send a total bitsize $O(n\phi)$ and whose bitsize of the PnS produced is in $O(|TSS|)$.

1.3.4 Extension: Handling External Validity

We illustrate in §4 and §5 how the techniques extend to the following case of *External Validity conditions* (Definition 13) on the inputs & output value. Recall that consensus may be used to provide a functionality denoted as "state machine replication" ([Lam98, CL99]). For this kind of usage, additional restriction may be placed on the possible output values. Namely, values are, in addition, to be returned as valid be some public algorithm. For instance, as bearing the signature of some accredited person, which may be denoted as a client. This is captured by the "External Validity Condition" of [CKPS01, Def 4.1]. It introduces an external "external validation entity" "Q_{ID}", that may send to players its signature on some values. These signatures are called "Certificates of validity". Players can forward these signatures to each other. We then say that a value is valid for a player, if this player has a Certificate of validity on this value In the External Validity setting, some players may not have initially an input that comes with a validity certificate. Also, players may possibly also have no input at all. Optimal Latency of leader-based consensus is then narrowed (Definition 14) to phases where: either a honest leader knows a valid value, or at least $t + 1$ honest players have a valid input (so the leader is guaranteed to be declared one). We handle this extra difficulty in §4 and §5.

As a slight alternative to these specifications of External Validity, notice that, in some use-cases (e.g. [CL99]), one could want to guarantee output in finite time in situations where t honest players or less would have a valid input. We formalize in Lemma 27 a possible add-on to protocols, without communication overhead, that guarantees the output of a default Nonvalid value in these cases (which then violates External Validity).

1.3.5 Extension: Preserving Optimal Latency Despite Interactive Proofs

In the baseline Figure 3 (from [GAG+19]), the leader is required to forward the set $\mathcal{D}$ of messages in **1**, immediatly after he receives it. To achieve Main Theorem 7, the leader is required to deliver a short proof (PoE) for the value v_L that he proposes, in place of this set $\mathcal{D}$. So the leader does not have the time to perform further interaction with players before he must issue this proof in **1**. Interestingly, a tweak on the baseline protocol 3, which we describe in 6, *enables* the leader to use 2 more messages delay to issue this proof, in **1**. We will leverage this extra delay in the alternative elementary *interactive* protocols, given in §5.1 and §5.2, which enable the leader to deliver a PoE in 4 messages delay. This tweak however specifies that the leader must anyway "commit" as soon as in **1**, in his proposes message, to the same value v_L for which he is going to deliver a proof in **3**.

Optimization for NonResponsive Protocols The tweak carries over unchanged to Interactive protocols for PoE which are NonResponsive (Definition 17). But then this makes the overall consensus NonResponsive. So while we are here, we chose the baseline NonReponsive consensus that offers an (a priori sharper) communication complexity, which is the one of [AGM18]. Indeed, this baseline does not require a PnS anymore while still having Optimal Latency. We recall this baseline in Figure 7, then apply the tweak to it in Figure 8.

1.3.6 Of Independent Interest: Elementary (Interactive) PoE and PnS

Weakening of Definition of PoE*.* To enable efficient elementary protocols, we will weaken the definition of a PoE(v) (from Equation (2) / Definition 11, into Definition 15). In particular it is not a proof of knowledge of a set $\mathcal{D}$ anymore. Anyway this weakened Definition of a PoE(v) still guarantees that no other value than v can be the input of all honest players. So is enough for the purpose of guaranteeing Strong Unanimity.

Elementary Responsive 4*-move* PoE*.* Then in §5.1 we construct a Responsive 4-move protocol that delivers PoE of size $O(\log(|\mathcal{V}|))$ threshold signatures ($\mathcal{V}$: the range of possible input values). *Applying* to the tweak on the *responsive* protocol of §2.1, as specified in §4.3.2 gives in addition Responsiveness with Optimal Latency.

Elementary NonResponsive 4*-move* PoE*.* In §5.2 we construct alternative short PoE consisting of 5 threshold signatures. But at the cost of a partially synchronous interactive 4-moves protocol with the signers. So which is NonResponsive. From which we recover nonetheless a Fast Track (in two messages delay) and Strong Unanimity in §4.3.3 with same tight latency of 6 rounds, thanks to a tweak to the baseline consensus of [AGM18].

All those elementary constructions of PoE also handle External Validity, whose Definition is recalled in §1.3.4.

An Elementary (Non Interactive) Protocol for Producing a PnS*,* is finally provided in 6. The protocol requires each player in phase number ϕ, to send a total $O(\phi)$ bits. The PnS output by the leader has size $|TSS|$.

1.3.7 Reducing the Latency of the state of the art *Leaderless Asynchronous* Validated Consensus ("VABA") of [AMS19].

Their protocol consists in running n instances of a variation of the consensus of [YMR$^+$19] in parallel, one for each player playing the role of a leader. In detail, our *phase* of the execution of the protocol is denoted there as *Proposal Promotion*, while the local variables denoted ϕ_i in [AMS19, Algorithms 5 6] correspond to our highest lock certificate seen by player i in our notations. Then players elect a leader a posteriori. The consensus of Theorem 7 can be used as a drop-in replacement for [YMR$^+$19]. This is detailed in 7.1. This results in a first phase in 5 messages delay (no Fast Track nor Strong Unanimity is needed in their case), instead of 7. Then is the first elected leader is dishonest or the network not initially fast enough, this results in higher phases in 6 messages delay instead of 8.

1.3.8 Amortization of Bit Complexity over Multiple (Ordered) Instances In Parallel.

In the use-case of consensus of [CL99], denoted as "state machine", players are executing an ordered sequence of consensus instances. But in case where many consecutive leaders do not enforce any output, then pending executions of these instances pile-up. As a result, a leader of some phase ϕ in Figure 2 at step **1**, receives not one set $\mathcal{R}$, but as many sets $\mathcal{R}_j$ as pending instances of consensus that the players are executing in parallel. This is handled in [CL99] by having the leader handle separately each instance in step **1** (even if he concatenates them in a single message for each player). As a result, applying Theorem 7 to this naive approach in M instances of PnS in parallel, would result in a total $O(Mn\log(n))$ bits sent. However, the proof system, e.g. the one of [AC20], being in logarithmic size in the circuit, this enables to compress M instances of proofs in a total bitsize $O(n\log(Mn))$. The only incompressible cost that remains is the public parameter v_{max} of each of these M instances, that need to be communicated to players, which represents a total constant size $O(M)$. The same applies to PoE.

1.3.9 Ensuring *Simultaneously*: Halting in Finite Time with Amortized Linear Complexity Over Long Values, *and* Requiring External Validity

Halting in Finite Time with Amortization over Long Values, by [NRS$^+$20]. Halting So far our protocols do not guarantee that players can *stop* playing protocol (this action being denoted as "Halting") in finite time after they all output. Only the latency to when he outputs is guaranteed. Guaranteeing halting would require that players who received a decision certificate, forward it to all other players, so they can stop taking part to the protocol. So these all to all messages bring us back to *quadratic* complexity in n, and linear in the bitlength $\ell := \log |\mathcal{V}|$ of the values. The general tool introduced [NRS$^+$20, Figure 4] compiles any consensus with output in finite time, into Halting with a cost can be amortized when ℓ is large, and turned into: $n^2 O(1) + n\ell$. So that when ℓ is large we recover our linear $n\ell$ complexity.

Incompatibility with External Validity However this compilation comes at a cost: if we are not in the situation where all honest players would have the same input, then this compiler may well force them to output some default value $\perp$. Rephrased in the notations of [NRS$^+$20, Figure 4]: a player is "happy" if value output by the baseline consensus (performed on hashes) is his input value. Then, if *some* honest players are not happy, the compiler *may arbitrarily* force them to output $\perp$. Technically, this is due to a subroutine of Consensus on being "happy" or not. But $\perp$ is not valid. Since this happens even if all players have a valid output, then, this *does not* satisfy the traditional "external validity" requirement of [CKPS01, Def 4.1] (or as in [AMS19, Lemma 23 of full paper]). Recall that this traditional requirement is that players output in finite time if they *all* start with valid inputs. As a remark, notice that these traditional requirements for output are themselves much stronger than our Definition 14. Nevertheless, the compiler is not compatible with these narrow traditional requirements.

Fortunately, it turns out that leader-based consensus protocols can escape this problem, thanks to a variation on the compiler of [NRS$^+$20] that we provide in §7.2.

1.3.10 Other Various Extensions and Consequences

Such as for the regime denoted as "state machine replication" (in a historical conception), are provided in §8.2. In particular we make explicit in §8.2.3 a more tolerant Fast Track in 3 messages delays (Definition 1), preserving latency of 5 in the first phase, by adapting [GAG$^+$19] with short proofs. We also briefly review how known optimization techniques apply.

1.4 Related work

Reponsiveness with Optimal Latency, Linear bit complexity in the worst case and Optimal adversary bound. [YMR$^+$19] is responsive, but at the cost of a worst case latency of 8 messages delay for a honest leader and timely phase, instead of the optimal 6, so does not satisfy Responsiveness with Optimal Latency.

Fast Track and/or Strong Unanimity. All consensus so far with either a Fast Track, or Strong Unanimity, have quadratic bit complexity of communications per phase : [Lam02, MA05, Kur02, KAD$^+$09, CKL$^+$09, AGK$^+$15, GV10, ANRS20]. To be sure, most of them are cast in the "state machine replication" setting, where the inputs are distributed in a preliminary step by the possibly malicious leader (or an external "proposer" or "client"). The complexity bottleneck is exactly the same of for Strong Unanimity described in §1.2.3 It happens when the leader forwards the set of declare messages $\mathcal{D}$ to each player. In the traditional terminology of Fast Tracks, in these declare messages the players testify the (fast) vote they cast in the first phase. But technically this plays exactly the same role as in §1.2.4.

Transparent setup means that no prior interaction between players nor trusted dealer of keys (as in [KSM20, Sho00, TCZ$^+$20]) is required before the protocol starts. Recall by contrast that modern efficient randomized consensus with subquadratic communication either require correctly generated secret keys ([ACD$^+$19, CPS19]) and/or a public random string (a.k.a. "genesis block") appearing *after* the keys are published ([DGKR18, CM19, CPS19, CKS20, BKZL20]). The only exception is the recent [BCG20a], under synchrony, which also uses proofs of knowledge of signatures

Synchronous setting The recent [MR20], under synchrony, achieves amortized linear communication complexity but has suboptimal adversary bound $t < n/2 - \epsilon$. Likewise, the synchronous [BCG20a] has suboptimal adversary bound $t < n/3 - \epsilon$.

1.5 Roadmap

In §2 we prove Theorem 7 from black box $\log(n)$ proofs "PnS" and "PoE", as announced. In §3 we recall and details to the model. We then consider extensions, and also the elementary constructions that support our claims in Theorem 8. From that point, we handle explicitely additional External Validity conditions on the inputs/output. In §4 we upgrade the previous Consensus of §2 to handle for PoE that may be obtained from an *interactive* proof protocol. We also redefine PoE with an alternative weaker definition, which is although still sufficient for the previous proof of Main Theorem 7. We introduce in §5 elementary interactive protocols constructing PoE in this weaker definition (a Responsive one, and NonResponsive one). Likewise in §6, we implement a weaker-but-sufficient version of PnS with an elementary protocol. Finally in §7 we detail more the other minor contributions. In §8 we provide some easy Lemmas and discuss other various extensions of the results.

2 Proof of Theorem 7: Consensus, From Black Box Proofs of Knowledge

2.1 Consensus with Responsiveness and Optimal Latency (Definitions 5 and 4) with overall $O(n\log(n)\log(V))$ Bits Sent per Phase

We define a drop-in proof of knowledge of the set $\mathcal{R}$, called "PnS", specified with the notations of the protocols of Figures 1 and 2 . We then use it in the aforementionned protocols. We merge their presentation into a single Figure 4, that both captures $\phi = 1$ and higher phases. Drop-in of the PnS is the only difference with the baseline Figures 1 and 2. We finally prove safety (and liveness) of the protocol with this drop-in proof.

2.1.1 Data Structures: Reminders and Complements.

Let us recall the definition of a *k-threshold signature scheme (TSS)* in the strong sense of [ACR20, §5]. Considering a baseline signature scheme, a k-threshold signature is a proof of knowledge of k signatures from distinct players on the same message content m. Here we consider a $(2t+1)$-TSS. We use the following data types:

- for a value $v \in \mathcal{V}$ and phase number $\phi \in [1, \ldots, \infty[$, we have messages of type $\mathsf{propose}(v, \phi, \mathsf{justifs})$, $\mathsf{lock\,vote}(v, \phi)$ and $\mathsf{decision\,vote}(v, \phi)$. The $\mathsf{justifs}$ are additional data, to be precised below, that are needed when $\phi \geq 2$ to make honest players accept the $\mathsf{propose}$ message as valid;
- $\mathsf{lock\,certificate}(v, \phi)$ is a $(2t+1)$-threshold signature on $\mathsf{lock\,vote}(v, \phi)$, while $\mathsf{decision\,certificate}(v)$ is a $(2t+1)$-threshold signature on $\mathsf{decision\,vote}(v, \phi)$ for *any* fixed ϕ;
- consider $0 \leq \phi_i \leq \phi$. We have messages of type $\mathsf{report}(\phi_i, \phi)$, such that $\phi_i \leq \phi$. If $0 < \phi_i$, then it is appended with a $\mathsf{lc}_i = \mathsf{lock\,certificate}(v_i, \phi_i)$ for some $v_i \in \mathcal{V}$;
- Our specialized definition for a *proof of Non-Supermajority* in this context, is a proof of knowledge of the following:

$$\text{(1)}\quad \mathsf{PnS}(\phi_{max}, \phi) = \Big\{ \Big(\text{Public Input}: \ \phi_{max} \leq \phi \ ; \ \text{I Know}: \mathcal{I} \subset \{1, \ldots, n\} \text{ of size } 2t+1 \\ \text{and } \mathcal{R} = \big\{ \ \mathsf{report}(\phi_j, \phi) \text{ signed by } P_j \ , \forall j \in \mathcal{I} \big\} \Big) \text{ Such That}: \big\{ \phi_j \leq \phi_{max} \text{ for all } j \in \mathcal{I} \big\} \Big\}$$

We denote accordingly the corresponding *existential* statement: we say that ϕ_{max} satisfies the *Non-Supermajority Predicate Up To* ϕ, if such a set $\mathcal{R}$ exists. So a $\mathsf{PnS}(\phi_{max}, \phi)$ proves in particular that ϕ_{max} satisfies the Non-Supermajority Predicate Up To ϕ. We denote $|\mathsf{PnS}|$ the size of a PnS.

- Finally, the $\mathsf{justifs}$ mentionned above are: a $\mathsf{PnS}(\phi_{max}, \phi)$, where $\phi_{max} = 0$ iff $v_{max} = \perp$, appended with: a $\mathsf{lc}_i = \mathsf{lock\,certificate}(v_{max}, \phi_{max})$ or $\perp$ iff $\phi_{max} = 0$.

2.1.2 Protocol and Proof

0 Report *If $\phi = 1$, then skip directly to step 1.* Every player P_i sets $\phi_i \leq \phi$ the highest phase up to ϕ for which he saw a $\mathsf{lock\,certificate}$, or $\phi_i = 0$ if he did not see any so far. He sends to the leader L_ϕ: a $\mathsf{report}(\phi_i, \phi)$, appended with a $\mathsf{lock\,certificate}(v_i, \phi_i)$ if $\phi_i \geq 1$.

1 Propose The leader L_ϕ: if $\phi = 1$, then he sets v equal to his own input and multicasts $\mathsf{propose}\big(v, 1, \perp\big)$. Otherwise, upon receiving valid report messages from $2t+1$ distinct players, then, letting ϕ_{max} be the highest ϕ_i received, builds a $\mathsf{PnS}(\phi_{max}, \phi)$ out of them.

(i) either $0 < \phi_{max}$, thus he received at least one $\mathsf{lc}_{max} := \mathsf{lock\,certificate}(v_{max}, \phi_{max})$. Then he multicasts $\mathsf{propose}\big(v_{max}, \phi, \big\{\mathsf{PnS}(\phi_{max}, \phi), \mathsf{lc}_{max}\big\}\big)$;

(ii) or if $0 = \phi_{max}$, he sets v equal to his own input and multicasts $\mathsf{propose}\big(v, \phi, \big\{\mathsf{PnS}(0, \phi), \perp\big\}\big)$.

2 Lock vote Every player P, upon receiving a valid $\mathsf{propose}\big(v, \phi, \mathsf{justifs}\big)$ from L_ϕ for the first time in ϕ, replies with a signed $\mathsf{lock\,vote}(v, \phi)$.

3 Lock certificate Leader L_ϕ, upon receiving $2t+1$ $\mathsf{lock\,votes}$ for the same (v, ϕ), issues from them a $\mathsf{lock\,certificate}(v, \phi)$, which he multicasts along with v.

4 Decision vote Every player, upon receiving from L_ϕ a $\mathsf{lock\,certificate}(v, \phi)$ for the first time in ϕ (whatever v), replies with a signed $\mathsf{decision\,vote}(v, \phi)$.

5 Decision certificate Leader L_ϕ, upon receiving $\mathsf{decision\,vote}(v, \phi)$ from $2t+1$ distinct players for the same value v, issues from them a $\mathsf{decision\,certificate}(v)$, that he multicasts to the players.

6 Output Upon receiving a $\mathsf{decision\,certificate}$ for v, a player $\mathsf{outputs}$ v.

Fig. 4. PBFT leader-centered: any phase ϕ, with bit complexity in $O(n\big|\mathsf{PnS}\big|)$

Proof of Communication Complexity. since honest leaders send to each player messages containing a constant number of threshold signatures and of PnS, we have that the worst case bit communication complexity per phase is in $O(n)$ times the max of the sizes $\big|\mathsf{PnS}\big|$ and of threshold signatures.

Optimal Latency. Results directly from the ability to build a PnS out of any set $\mathcal{R}$ of $2t+1$ well formed report messages.

Proposition 9. *The protocol of Figure 4 has* Consistency.

Preliminary remark: no two distinct lock certificate relative to the same phase can be formed. Indeed honest players send at most one lock vote per phase.

Proof. Consider an execution in which some value v is output by some player. Thus a $\mathsf{decision\,certificate}(v)$ has been formed in this execution. Thus there exists a phase ϕ_v such that $2t+1$ distinct players in this phase signed a $\mathsf{decision\,vote}(v, \phi_v)$. Without restricting generality, one can ϕ_v is minimal with this respect. Suppose that another value v' is output by another player in the same execution. Then there exists a phase ϕ' such that $2t+1$ distinct players in this phase signed a $\mathsf{decision\,vote}(v', \phi')$. By the minimality assumtion, we have $\phi_v \leq \phi'$. By the preliminary remark, we thus have $\phi_v < \phi'$. But this is impossible, by Lemma 10. ☐

It remains to prove the following key invariant, which also holds in [CL99].

Lemma 10. *Suppose that there exists a phase ϕ_v in which $2t+1$ $\mathsf{decision\,vote}(v, \phi_v)$ are sent for some value v. Then there does not exist any other value $v' \neq v$ and higher phase $\phi_v < \phi'$ in which some honest player sends a $\mathsf{lock\,vote}(v', \phi')$ for v'.*

Proof. Now, suppose by contradiction that such a higher phase $\phi_v < \phi'$ exists, and that it is *minimal* with this respect. That is: ϕ' is the *smallest phase number after* ϕ_v: $\phi_v < \phi'$ in which some honest player sends a lock vote for a conflicting value $v' \neq v$. This would mean that this honest player receives in ϕ': a $\mathsf{PnS}(\phi_{lc'}, \phi')$ for some $\phi_v < \phi_{lc'} < \phi'$, along with a lock certificate$(v', \phi_{lc'})$.

(i) On the one hand, we cannot have $\phi_v < \phi_{lc'} (< \phi')$. Indeed otherwise, we would have that: a lock certificate is formed in phase $\phi_{lc'}$ for the value v' that *conflicts* with v, and thus that a lock vote *conflicting* with v is sent by some honest player in $\phi_{lc'}$. So this contradicts the minimality of ϕ' with this respect.

(ii) So we must have that $\phi_{lc'} < \phi_v (< \phi')$. But then, existence of this $\mathsf{PnS}(\phi_{lc'}, \phi')$, proves by definition that *no* possible quorum of $t+1$ honest players did observe, *before or while they were in phase* ϕ', some lock certificate for some locked value (v_1, lc_1) conflicting with v': $v_1 \neq v'$, relative to some higher phase ϕ_1: $\phi_{lc'} < \phi_1 < \phi'$. So a fortiori there does not exist any higher phase ϕ_1: $\phi_{lc'} < \phi_1 < \phi'$ in which some honest player could cast some decision vote for a conflicting $v_1 \neq v'$. But this precisely contradicts the assumption on $(v_1 := v, \phi_1 := \phi_v)$. □

2.2 Adding the Fast Track and Strong Unanimity

We now solve the quadratic bottleneck for Strong Unanimity. As we saw in §1.2.4 (or Lemma 24 below), this will then also solve for free the one of a Fast Track. As in §1.2.3 we introduce the new message type declare(v). Each (honest) player is asked to issue signed declare(v) messages only for his input value v. Our specialized definition for $\mathsf{PoE}(v)$ in this context is a proof of knowledge:

$$(2) \quad \mathsf{PoE}(v) = \Big\{ \Big(v \ ; \ \text{I Know}: \ \mathcal{I} \subset \{1, \ldots, n\} \text{ of size } 2t+1, \ \mathsf{and}\ \mathcal{D} = \big\{ \mathsf{declare}(v_j) \text{ signed by } P_j \ , \forall j \in \mathcal{I} \big\} \Big) \\ \text{Such That}: \ \text{ no value } v' \neq v \text{ is such that } \big\{ v_j = v' \text{ for } t+1 \text{ or more indices } j \in \mathcal{I} \big\} \Big\}$$

We denote accordingly the following corresponding *existential* statement:

Definition 11. We say that a value v_{un} satisfies the *Exclusivity Predicate* with respect to some execution of the protocol, if and only if, there exists a set $\mathcal{D}$ of $2t+1$ declare(v_i) messages issued by distinct players in this execution, such that no value $v' \neq v$ repeats identically more than $t+1$ times in these $2t+1$ declared input values v_i.

So a $\mathsf{PoE}(v)$ proves in particular that v satisfies the Exclusivity Predicate. Which in turns guarantees that v is the only possibly unanimous input of all honest players (if any), by the easy Lemma 25. Given this black box, the protocol of Figure 4 needs only be enriched as in Figure 3, using $\mathsf{PoE}(v_L)$ as a drop-in replacement for exhibiting the set $\mathcal{D}$.

0 Report *unchanged*

1 Propose The leader L_ϕ, upon receiving a set $\mathcal{R}$ of report messages from $2t+1$ distinct players, then, letting ϕ_{max} be the highest ϕ_j in $\mathcal{R}$:

- (i) *either* $0 < \phi_{max}$, *then as in Figure 4* ***1(i)***
- (ii) or: let $\mathcal{D}$ be the set of $2t+1$ declare messages received. The leader selects any value v_L reported in $\mathcal{D}$, such that *no conflicting value* $v' \neq v_L$ is reported identically in $t+1$ messages of $\mathcal{D}$. He constructs a $\mathsf{PoE}(v_L)$ out of $\mathcal{D}$. Then he multicasts propose$\big(v_L, \phi\big)$ appended with the justification $\mathsf{PoE}(v_L)$ (in addition to the justification $\mathsf{PnS}(0, \phi)$, as in Figure 4).

2 Lock vote Is enriched in case (ii): in addition to the previous check on $\mathcal{R}$, every player also checks validity of the $\mathsf{PoE}(v)$ for the value v proposed.
(Then if this checks also passes, he replies with a signed lock vote(v, ϕ), as in Figure 4)

Fig. 5. Adding Strong Unanimity with $O(n\big|\mathsf{PoE}\big|)$ complexity

Proposition 12. *The Consensus of Figure 5 satisfies, in addition, Strong Unanimity.*

Proof. Suppose by contradiction that all honest players have (unanimously) the same input v_{un}. Consider any value $v' \neq v_{un}$, let us show that no $\mathsf{lock\,certificate}(v', \phi')$ can be formed in the execution (and a fortiori no $\mathsf{decision\,certificate}(v')$). Without loss of generality, let us consider ϕ' the smallest phase in the execution where a $\mathsf{lock\,certificate}(v', \phi')$ is formed for a value v' different from v_{un}. Thus at least $t + 1$ honest players sends a $\mathsf{lock\,vote}(v', \phi')$ in **2** of phase ϕ'.

Claim 1: None of these honest players can be in situation (ii). *Proof of claim 1* if such a honest player is in situation (ii) of **2**, then this means he received from the leader a valid $\mathsf{PoE}(v')$. So by the easy Lemma 25 this violates unanimity of v_{un}.

Continuation of the Proof Thus all these $t + 1$ honest players are in case (i). Thus they received from the leader a $\mathsf{lock\,vote}(v', \phi_1)$ formed in a phase lower or equal $\phi_1 \leq \phi$.

Claim 2: So we have that $\phi_1 < \phi$. Indeed, otherwise, the $\mathsf{lock\,certificate}(v', \phi')$ would have been formed without the $\mathsf{lock\,votes}$ of these players, which is impossible.

End of the Proof This contradicts minimality of ϕ'. □

3 Model: Reminders and Complements

3.1 Reminders

Static malicious (Byzantine) corruptions, with authentication We consider $n = 3t + 1$ players, which are polynomial machines with fixed public keys issued by them. They send messages to each other that carry their digital signatures ([Bol03]), which are thus assumed unforgeable. But the content is read in clear by the Environment. Each player is being given an input value v in some finite range $\mathcal{V} = [0, \dots, V]$ by the Environment. For instance, values may be of the size of a hash: $V = 2^{256} - 1$. Up to t players are corrupted by the polynomial Environment and behave as instructed by him: they are called *malicious players.* Remaining players are called “honest”.

Synchronizing phases and leaders There is an entity called “Synchronizer” (in [NBMS19, NK20, BCG20b], a.k.a. “Pacemaker” in [YMR+19]) which has a local counter: ϕ, called “phase” (also known as view/epoch/polka or even “round” used in a nonstandard way), which increases monotonically at various speed. Every time the local counter ϕ increases, then the Pacemaker sends messages to players indicating the new value of ϕ, along with the identity of (possibly new) specific “leader” player: L_ϕ. Messages notifying this being sent on the fully asynchronous network, they arrive with arbitrary delays to players. We say that a player “is in phase ϕ” if ϕ is the highest phase that he was notified of so far.

Worst-case bit complexity of communications We count the total number of bits sent by all honest players in the same phase. Then take the worst case over all phases, over all executions. Notice that under some weaker synchronizers than our “Pacemaker”, many players may possibly believe that they are the leader of the same phase ϕ. In these cases we would then multiply complexity by the number of those players. But for simplicity we do not consider such models.

3.2 Messages lost or not, output vs Halting

For Simplicity: Messages are Always Delivered. Recall that Definiton of *Optimal Latency* (Definition 4) assumes for simplicity an asynchronous network where messages *cannot* be lost. Main Theorem 7 and Theorem 8 (the Responsive claim) are stated with respect to this Definition 4 in this asynchronous model.

Comment: Adaptations to the Model if Messages Could be Lost. We could alternatively have supposed that messages sent can be lost, and restricted *Optimal Latency* (Definition 4) to phases where no message is lost. The proofs of Main Theorem 7 and Theorem 8 hold unchanged in such an alternative model.

Halting. Recall that Definition 4, latency is considered only until players output. However we *do not* guarantee that they can halt in finite time. *In our fully asynchronous where messages are never lost*, then one could enforce halting by having a player which outputs to propagate the decision certificate to all players then halt. However these additional instructions trigger *quadratic* message complexity. This is why we do not consider halting neither in Main Theorem 7 nor Theorem 8. However we *will* consider halting in §7.2, and consider the bit comlexity amortized over long values. But we will so only in models in which messages *cannot* be lost. Indeed, in models in which messages can be lost, then achieving halting in finite time is anyway *not* possible by [DLS88, §4.2 Remark 2].

3.3 Partial Synchrony.

For the specific case of the last NonResponsive claim of Theorem 8, on Nonresponsive protocols (supported by the protocols §4.3.3 and §5.2), we will depart from our fully asynchronous model and consider instead partial synchrony. This assumes a publicly known delay Δ, and some unknown time GST, after which messages are delivered within Δ.

Messages can be Lost. All the partially protocols considered (in §4.3.3 and §5.2) hold even under the specific model [DLS88, §3.1] where messages can be lost before GST. Thus also our NonResponsive claim of Theorem 8.

Except if halting is required. However, to apply the amortized halting techniques of §7.2, then, as mentionned in 3.2, one needs to assume in addition that messages are never lost, such as in the model [DLS88, 2.3 3)].

4 Handling External Validity and PoE with Interactions

4.1 Handling for External Validity Conditions

In the remainder of this paper we make the choice to generalize a bit the constructions in order to handle Consensus with External validity, as defined in [CKPS01, Def 4.1]. That is, we assume that an external entity Q_{ID} may deliver to some honest players "certificates of validity" of their inputs values (that they can forward). We say that a value is valid *relatively to a player and some point in time*, if at this point of time, this player has a certificate of validity for this value. We accordingly restrict the Definition 1 of Consensus to handle External Validity conditions on the output:

Definition 13 (Consensus with External Validity)**.** **- Consistency:** *unchanged*

- **External Validity:** Players do not output a value which is not valid
- **Consensus Weak Unanimity (CWU):** *If all n* players are honest *and* all start with the same initially valid input; then, this is the only value that they can possibly output;
- **Consensus Strong Unanimity (CSU):** *If all honest players* start with the same initially valid input; then, this is the only value that they can possibly output;

Notice that loosening the CWU and CSU enables players to output some value even if all honest players have the same Nonvalid input. We will explain after Definition 15 how tolerating this loosening enables in return *better liveness in protocols* (Definition 14), compared to in the traditional leaderless External Validity conceptions ([CKPS01, Def 4.1] or as in [AMS19, Lemma 23 of full paper]). To guarantee External Validity, all Consensus protocols may be updated by requiring players to ignore any message containing a value not valid. This is formalized by the easy Lemma 26. In practical use cases, players may possibly gossip validity certificates they received. Players encode their input by 0 if it is not valid, i.e., if they have not seen so far a validity certificate for it. They also encode it as 0 if they have no input at all.

Further Comments on Definition 4.1. The behavior of Q_{ID} is arbitrary, it may possibly issue certificates only for a subset of input values. The Lemma 27 provides a path, with linear complexity, to output a default value in finite time when $\leq t$ honest players have a valid input. But allowing this path comes at the cost of possibly violating *External Validity.*

Restriction of output guarantee (Liveness) under External Validity.

Definition 14 (Optimal Latency)**.** The additional restriction is made to Definition 4 that output is guaranteed in a phase only if this phase further satisfies: *Either* the (honest) leader of this phase knows a valid value (which holds in particular if his input is valid) *Or* at least $t + 1$ honest players have a valid input.

Notice that this Liveness Definition is more demanding than the traditional requirement ([CKPS01, Def 4.1] or as in [AMS19, Lemma 23 of full paper]), where output is *not* guaranteed as soon as one player does not have a valid input. For use cases where an output would be anyway required when the above conditions (honest leader valid input of $t + 1$ honest players valid input) are not satisfied, then Lemma 27 brings a mechanism for output *under no additional condition on the inputs* (so under plain Definition 4), but at the cost of possibly violating External Validity when t honest players or less have a valid input.

4.2 Useful Weakening of Exclusivity Predicate. Extension with the Special Nonvalid Value

For our Elementary constructions in §5, we define a new Exclusivity Predicate which is weaker than (or equal to) the previous one of (2). This new Definition, which we denote *Exclusivity-uni*, is the property implied from Equation (2) by Lemma 25. Namely, *a value in $\mathcal{D}$ satisfies* Exclusivity-uni *if no other value is a unanimous input of honest players.* However, this weakened (or equal) new Definition of still states exactly the property of PoE which is used in the proof of Proposition 12. So we can still use PoE satisfying this new Definition in the protocols. In addition, we also extend this new (a priori weakened) definition to handle the Nonvalid 0 value, in addition to values in $\mathcal{V} = [1, \ldots, V]$. This results in the following more intrinsic Definition, that will be of particular help for handling *negative* statements of players.

Definition 15 (PoE)**.** We say that $v \in \{0\} \cup \mathcal{V}$ satisfies the Exclusivity Predicate, if and only if, no other value $v' \in \mathcal{V}$ is a unanimous input of honest players. A $\mathsf{PoE}(v)$ is a proof that v satisfies the Exclusivity Predicate.

This Definition is none other than the previously mentionned *Exclusivity-uni*, extended to 0. For simplicity we dropped the terminology *Exclusivity-uni.* Indeed we do not consider the previous one (Equation (2)) anymore in the subsequent elementary methods.

Motivation of Definition 15, in link External Validity (Definition 13) Definition 15 concretely enables a leader who would receive $2t$ declarations of having 0 input, to nevertheless issue a $\mathsf{PoE}(v_L)$ on his own (valid) input. This tolerance in Definition 15 is the key to achieving better liveness in our protocols (Definition 14), than demanded traditionally ([CKPS01, Def 4.1] or as in [AMS19, Lemma 23 of full paper]). Notice that this new Definition 15 *coincides* on $\mathcal{V}$ with the previously mentionned *Exclusivity-uni.* Precisely, the (iii) in the Lemma below points that we can well have that the value 0 is a *unanimous* input of all honest players, *and still*, this implies that *every* value $v \in \mathcal{V}$ also satisfies the Exclusivity Predicate. Protocols below will indeed allow this situation where a $\mathsf{PoE}(v)$ can be formed for $v \neq 0$ and v output, even if 0 is unanimous among honest players. Indeed, tolerating this situation does not contradict anymore Weak Unanimity nor Strong Unanimity, as weakened in the new Definition 13.

Lemma 16. *(i) Let $v \in \{0\} \cup \mathcal{V} = [0, \ldots, V]$ be any value. The data, for each value $v' \neq v$, of $t + 1$ signed messages from distinct players stating that they* do not *have input v' constitutes a PoE(v).*
(ii) Let $v \in \{0\} \cup \mathcal{V}$ be any value. The data of $t + 1$ signed messages from distinct players stating that they have input v constitutes a PoE(v).
(iii) A PoE(0) constitutes a PoE(any): that is, that all values in *$[0, \ldots, V]$ satisfy Exclusivity (Definition 15)*

(iv) A PoE(w) along with $t+1$ statements of not *having input w, constitutes a PoE(any).*

Proof. (i): the data implies that for each value $v' \neq v$ in $\mathcal{V}$, there is at least one honest player who does not have v' as input, so v' cannot be unanimous among honest players. (ii): follows from (i): $t+1$ declarations to have input v also constitute in particular, for every other $v' \neq v$, $t+1$ declarations of *not* having input v'. (iii): Apply (i) to every $v \in \mathcal{V}$. (iv) the second data proves that w is not an unanimous input of honest players, but the first data proves that w is the only possible unanimous honest input. So finally *no* value can be a unanimous input. Thus *all* values satisfy the exclusivity predicate. □

4.3 Handling for *Interactive* 4-move PoE in the Consensus of Main Theorem 7

4.3.1 PoE with 4-move Interaction

We still consider $n = 3t+1$ players, of which t are maliciously corrupted. We denote L a distinguished player which we call "prover". Each player (which includes the prover) starts with an input value $v_i \in \{0\} \cup \mathcal{V} = [0, \ldots, V]$ with respect to this protocol.

Definition 17. We call a 4-*move interactive Responsive* PoE *protocol* a protocol between L and the players (including himself), such that if L is honest and: *If Either* L knows a valid value *Or* at least $t+1$ honest players have a valid input, then:

- After a round-trip of messages, L outputs a valid value v. We denote this select(v).
- Then after another round-trip he outputs a PoE(v) in the sense of Definition 15.

We call a 4-*move interactive Not Responsive* PoE *protocol*, in the model [DLS88, 3.1], a protocol played round-by round, and such that the previous properties hold only if it is initiated after GST (that is: if synchrony holds).

Notice that this implies by construction that the value v that the prover selects, then for which he outputs a PoE(v), satifies the Exclusivity Predicate (Definition 15).

4.3.2 Adding the Fast Track and Strong Unanimity to Consensus of Figure 4: Figure 6

Overview of the differences with Figure 5. Recall that obtaining Strong Unanimity (and thus subsequently Fast Track by Lemma 24) in Main Theorem 7, follows from enrichening the baseline Consensus of Figure 4, with a PoE, as specified in Figure 5.

However to use an *interactive* 4-move PoE protocol (Definition 17), then the leader cannot deliver anymore a PoE when at **1**(ii) of Figure 4. Thus we make the following alternative enrichening instead. If the leader is in case (ii) of **1**, then he selects the value v_L from the ongoing PoE protocol (Definition 17). He then propose(v_L), but is *not anymore* required to append a PoE(v_L) to it. Players then accept a propose(v) at **2**(ii) without anymore checking if a PoE for it is attached. This leaves the leader two more rounds to issue a PoE for the value v_L he proposed. This delay corresponds to the two last moves of the interactive 4-move PoE protocol (Definition 17).

The leader then outputs a PoE at **3**, which he appends to the lock certificate(v). Players now accept a lock certificate(v) in **4** *only* if it comes *appended* with a PoE(v_L). Likewise, players now report in **0** *only* lock certificate(v_i) that come *appended* with a PoE(v_i). Subsequently, the leader now takes in consideration reported lock certificate(v_i) in **1** *only* if they come appended with a PoE(v_i).

Formalization: Figure 6. The above additional validity requirement for a lock certificate is incorporated in the new enriched data structure which we denote as full lock certificate. A full lock certificate(v, ϕ) is the concatenation of: a $(2t+1)$ threshold signature on messages lock vote(v, ϕ) (which is what was denoted as lock certificate(v, ϕ)), *and* of a PoE(v).

0 Report Every player P_i sets $\phi_i \leq \phi$ the highest phase up to ϕ for which he saw a $\mathsf{lock\,certificate}$, or $\phi_i = 0$ if he did not see any so far. He sends to the leader L_ϕ: a $\mathsf{report}(\phi_i, \phi)$, appended with a $\mathsf{full\,lock\,certificate}(v_i, \phi_i)$ if $\phi_i \geq 1$. He *Initiates* an instance of a 4-move PoE (Definition 17) with prover L_ϕ, with respect to his input value.

1 Propose The leader L_ϕ, upon receiving a set $\mathcal{R}$ of report messages from $2t+1$ distinct players, then, letting ϕ_{max} be the highest ϕ_j in $\mathcal{R}$:

(i) either $0 < \phi_{max}$, thus he received at least one $\mathsf{flc}_{max} := \mathsf{full\,lock\,certificate}(v_{max}, \phi_{max})$. Then he multicasts $\mathsf{propose}\big(v_{max}, \phi, \big\{\mathsf{PnS}(\phi_{max}, \phi), \mathsf{flc}_{max}\big\}\big)$;
(ii) or if $0 = \phi_{max}$, then: *If* he selects a (valid) value v_{select} with respect to the ongoing 4-move PoE, then he multicasts $\mathsf{propose}\big(v_{select}, \phi, \big\{\mathsf{PnS}(0, \phi), \perp\big\}\big)$; *else* does nothing.

2 Lock vote Every player P, upon receiving a valid $\mathsf{propose}\big(v, \phi, \mathsf{justifs}\big)$ from L_ϕ for the first time in ϕ, replies with a signed $\mathsf{lock\,vote}(v, \phi)$.

3 Lock certificate Leader L_ϕ, upon receiving $2t+1$ lock votes for the same (v, ϕ),

(i) if L already had a full lock certificate on v when he proposed it in **1**, then he extracts from it the $\mathsf{PoE}(v)$;
(ii) *Else* this means that he selected v with respect to the PoE protocol, in **1**. Thus, by Definition 17, he now obtains a $\mathsf{PoE}(v)$.

He AGGREGATES the $2t+1$ lock vote (v, ϕ), and appends them with the $\mathsf{PoE}(v)$ obtained in (i) or (ii) above, into full lock certificate (v, ϕ) that he sends to the players.

4 Decision vote Every player, upon receiving from L_ϕ a $\mathsf{full\,lock\,certificate}(v, \phi)$ for the first time in ϕ (whatever v), replies with a signed $\mathsf{decision\,vote}(v, \phi)$.

5 Decision certificate Leader L_ϕ, upon receiving $\mathsf{decision\,vote}(v, \phi)$ from $2t+1$ distinct players for the same value v, issues from them a $\mathsf{decision\,certificate}(v)$, that he multicasts to the players.

6 Output Upon receiving a decision certificate for v, a player outputs v (but continues the protocol)

Fig. 6. Adding Strong Unanimity to Figure 4 with 4-move *Interactive* PoE

Sketch proof of this alternative enrichening. Responsiveness with Optimal Latency. If in case **1**(ii) then it is guaranteed by Definition 17 that, as soon as the leader has a valid input or $t+1$ honest players have a valid input, then, the leader will be able to select a valid v_{select} for which he will issue a PoE at step **3**, and thus produce a decision certificate for this value. If in case **1**(i), then it is the case that the leader obtained a set $\mathcal{R}$ of report messages containing at least one full lock certificate. Let full lock certificate(v_{max}, ϕ_{max}) be the highest, which he proposes to players, it is then guaranteed that the players will accept it thanks to the PnS certifying that ϕ_{max}) as the highest phase reported in $\mathcal{R}$. The leader can then subsequently issue a lock certificate then a decision certificate.

Consistency is unchanged, since the baseline Figure 4 is unchanged.

Strong Unanimity Results from a adaptation of the proof of Proposition 12. Namely: replacing lock vote(ϕ', v') by decision vote(ϕ', v') in the proof, then shows that we have the guarantee that a decision certificate(v), is formed only if v satisfies the *Exclusivity Predicate*. In the context of the new weakened Definition 15, then this directly implies that Strong Unanimity holds. [Otherwise if this is a PoE for the old Definition of Equation 2, then this implies a fortioti the new Definition 15 by Lemma 25.]

4.3.3 NonResponsive Consensus with Strong Unanimity Used with Interactive Proofs: Fig. 8

The Baseline NonResponsive consensus of [AGM18]: Figure 7 We still consider a $(2t+1)$-TSS. We consider the model of [DLS88, 3.1] with parameter Δ.

0 Report *If $\phi = 1$, then skip directly to step **1**.*

Otherwise: Every player P_i sets $\phi_i \leq \phi$ the highest phase up to ϕ for which he saw a lock certificate, or $\phi_i = 0$ if he did not see any so far. He sends to the leader L_ϕ: a report(ϕ_i, ϕ), appended with a lock certificate(v_i, ϕ_i) if $\phi_i \geq 1$. Leader L_ϕ *Waits for delay Δ.*

1 Propose Leader L_ϕ *waits* that Δ finishes to elapse, unless $\phi = 1$. He receives his messages.

- (i) If he ever observed so far a lock certificate (w, ϕ'), then he chooses the one with the highest phase number ϕ_{max} and multicasts propose (v_{max}, ϕ) for the corresponding value v_{max}, appended with the lock certificate (v_{max}, ϕ_{max}))
- (ii) else if he ever saw a valid value, then he chooses one v_L which he multicasts propose(v_L, ϕ). (*Else*, he does nothing.)

2 Lock vote Every player P, when in phase ϕ and upon receiving a propose (v, ϕ) from L_ϕ, possibly appended with a lock certificate (v, ϕ_v), then P accepts it if he has not seen any other lock certificate for a different w with a higher phase number $\phi_w > \phi_v$ (where by convention $\phi_v := 0$ if no lock certificate is appended). In which case P sends a signed lock vote (v, ϕ) to L_ϕ.

3 Lock certificate Upon receiving $2t+1$ lock votes for the same (v, ϕ), the leader L_ϕ.

4 Decision vote When in phase ϕ and upon receiving a lock certificate (v, ϕ) from leader L, a player P answers by a signed decision vote (v, ϕ_P).

5 Decision certificate Leader L_ϕ, upon receiving $2t+1$ decision votes for the same (v, ϕ), AGGREGATES them into a decision certificate (v, ϕ), that he multicasts.

6 Output Upon receiving a decision certificate for v, a player outputs v (and continues the protocol).

Fig. 7. Nonresponsive consensus of [AGM18]

Adding Strong Unanimity using Interactive PoE*: : Figure 8* We enrich Figure 7 with Strong Unanimity, with exactly the same tweak (2-message delay before delivering PoE) as in Figure 6.

The only addition to the proof is on Optimal Latency, and consists in noticing that, after GST, then a leader under the conditions of Definition 17 is indeed guaranteed to select in **1**, and to issue a PoE in **3**.

0 Report Every player:

(i) *Initiates* an instance of a (NonResponsive) 4-move PoE (Definition 17) with prover L_ϕ, with respect to his input value.
(ii) If any, sends to L_ϕ the full lock certificate (w, ϕ') on a value w with the highest phase number ϕ' that he received so far;

1 Propose Leader L_ϕ:

(i) If he ever observed so far a full lock certificate (w, ϕ'), then he picks the one with the highest phase number ϕ_{max}, and multicasts propose(v_{max}, ϕ) for the corresponding value v_{max}, appended with the full lock certificate (v_{max}, ϕ_{max})
(ii) *Else, if* in the ongoing execution of the PoE protocol, L_ϕ selects a (valid) value v_{select}, then he multicasts propose(v_{select}, ϕ); Else does nothing.

2 Lock vote Every player P, when in phase ϕ and upon receiving a propose(v, ϕ) from L_ϕ, possibly appended with a full lock certificate (v, ϕ_v), then P accepts it if he has not seen any other full lock certificate for a different w with a higher phase number $\phi_w > \phi_v$ (where by convention $\phi_v := 0$ if no full lock certificate is appended). In which case P sends a lock vote (v, ϕ) to L_ϕ, signed with τ.

3 Full lock certificate Upon receiving $2t + 1$ lock vote for the same (v, ϕ), the leader L_ϕ:

(i) *If* L already had a full lock certificate on v when he proposed it in **1**, then he extracts from it the $PoE(v)$;
(ii) *Else* this means that he selected v with respect to the PoE protocol, in **1**. Thus, by Definition 17, he now obtains a PoE(v) from this protocol.

He AGGREGATES the $2t + 1$ lock vote (v, ϕ), and appends them with the $PoE(v)$ obtained in (i) or (ii) above, into full lock certificate (v, ϕ) that he sends to the players.

4 Decision vote When in phase ϕ and upon receiving a full lock certificate $(v, \phi), PoE(v))$ from leader L, a player P answers by a decision vote (v, ϕ_P) signed with τ_P.

5 Decision certificate leader L_ϕ, upon receiving $2t + 1$ decision vote (v, ϕ) for the same value and the current phase, aggregates them with τ into a decision certificate (v, ϕ), that he forwards to the players.

6 Output Upon receiving a decision certificate for v, a player outputs v (and continues the protocol).

Fig. 8. Partially synchronous consensus with Strong Unanimity using Interactive PoE

5 Elementary interactive PoE

5.1 An elementary interactive Responsive PoE of bitsize $O(\log(V)|TSS|)$

5.1.1 Warmup in the binary case.

If the set of inputs is binary: $\{0\} \cup \mathcal{V} = \{0, 1, 2\}$, then the following is a responsive PoE protocol in one step between players and a prover L. Let us consider a threshold signature τ_P with threshold $t + 1$.

0 Players send to L: their input x_i, with the validity vertificate $x_i \neq 0$, along with testifies signed with τ: "my input is *not* 1", resp, "my input is *not* 2", and, accordingly, *both* testifies when they have input 0.

1 Upon receiving $2t + 1$ well-formed messages, then it must be that L received at least more than $t + 1$ copies of one of those testifies. Without loss of generality, suppose that he received $t + 1$ copies for *not* having the input 1. He AGGREGATES the signatures with τ into what we denote as $\neg 1$.

(i) *If* he received 2, then by well-formedness of the message, this came with a validity certificate for 2, thus 2 is valid. Also, $\neg 1$ constitutes a PoE(2) by Lemma 16 (i). The prover then outputs 2 and this PoE(2).

(ii) *Else* if he received no 2, then it must be the case that he received $2t + 1$ testifies of not having input 2. Which we denote $\neg 2$. Along with $\neg 1$, this constitutes a PoE(*any*). Thus as soon as the prover knows a valid value (possibly from receiving it from the players, which happens if at least $t + 1$ honest players have a valid input), then, let v_L be such a value, he outputs v_L.

5.1.2 Protocol and Proof.

Let us consider a threshold signature τ_P with threshold $t + 1$.

1 Report Each player P sends to L his value v_P signed with τ_P, along with a validity certificate if it is not 0; plus, for each bit b_i of his value v_P, one testify: "my i-th bit is equal to $\mathsf{b_i}$", signed with τ_P.

2 Early termination *or* Select and Request Upon receiving $2t + 1$ well formed messages, the prover L:

(i) If he received more than $t + 1$ times the same value v, then he AGGREGATES those reports. This constitutes a PoE(v) by Lemma 16 (ii).

(α) *either* v is valid, in which case he outputs v and the PoE(v);

(β) *or* v is zero, in which case the PoE(v) constitutes a PoE(*any*) by Lemma 16(iii). Thus as soon as L knows a valid value (possibly from receiving it from the players, which happens if at least $t + 1$ honest players have a valid input), then, let v_L be such a value, he outputs v_L and the PoE(*any*).

(ii) Otherwise he constructs the "Frankenstein" value $w = (b_i)$, such that for every $i \in [1 \dots \log V]$, the i-th bit b_i of w is the one which received the majority (so $\geq t + 1$) of testifies. He AGGREGATES each of those majority declarations, with τ, to obtain $\log V$ threshold signatures: one for every bit b_i of w. *Claim2*: by Lemma 16(i), this constitutes a PoE(w).

(α) If w is a valid value which is reported in one of the $2t + 1$ messages, then he directly outputs w, the validity predicate and the previous PoE(w) then terminates. this constitutes a PoE(w). In particular if $w = 0$, then Lemma 16(ii) shows that this then constitutes a PoE(*any*).

(β) *Else*, as soon as L knows a valid value (possibly from receiving it from the players, which happens if at least $t + 1$ honest players have a valid input), then, let v_L be such a value, he select(v_L) and sends to the players the question: "is w your input value ?"

3 Answer Upon receiving the question, each player P replies to L with the message "my input value is (resp. is not) w" and signs it with τ_P.

4 Output Upon receiving $2t + 1$ answers of players which *do not* contradict what they reported in **1**. We have that *Claim4*: at least $t + 1$ of those answers say *not* to have input w. L then AGGREGATES the signatures of these $t + 1$ negative answers. By Lemma 16(iv), the data of these $t + 1$ negative answers, along with the previous PoE(w), constitute a PoE(*any*). L then outputs v_L, the validity certificate on v_L and the PoE(*any*).

Proposition 18. *This is a 4-move interactive* PoE *protocol (Definition 17), with bit complexity of communication $O(\log V)$ and producing a* PoE *of size $O(\log V|TSS|)$.*

Proof. The communication complexity is straightforward, and the size of the PoE output in **4** consists in: $\log(V)$ aggregated signatures on bits, plus one aggregated signature of $t+1$ players declaring not to have input w.

Let us now prove that the protocol satisfies Definition 17. We do so by justifying the inline claims in the protocol that were not yet proven.

We first prove *Claim2* made in **2**(ii)(α), which is a generalization of the previous warmup. For every value $w' \neq w$, we have that there exists a bit i at which w and w' differ. The i-th aggregated testify then proves that b_i was present in the input of more than $t+1$ players. So in turn this proves that the i-th bit of w': $1-b_i$, *cannot* be present in $2t+1$ input values or more.

Let us finally justify the *Claim4* made in **4**, which will conclude our proof. What is to be shown is that, among the $2t+1$ noncontradictory declarations received in **4**, we have *at most* t of them which can claim to *have* input w. But remember that *none* of the $2t+1$ players reporting in **1** declared to have input w: indeed, otherwise, the prover would have already terminated by **2**(ii)(α). So the *only* players who can report having input w in **3** *without* contradicting their Report received in **2**, are precisely those whose report was simply *not* received by the prover in **2**. Since we have no more than t of such players, this is what was to be shown. □

5.2 A three-move PoE under partial synchrony with size in $6|TSS|$

Let v_{max} be the highest input of honest players.

Theorem 19. *The following protocol is a partially synchronous 4-move PoE protocol with overall $\log(n)\log(v_{max})$ communication, and the* PoE *output has constant size, in $\log(v_{max})$).*

Let $\tau = (\tau_P)$ be a $t+1$ threshold signature scheme. The idea of the protocol is inspired from group testing [DH93]: after receiving reported input values, the prover L sorts them in consecutive intervals I_j containing few enough reported values. He then asks players to *declare* in which set I_j their values are *not*. Thus, unless we are in the easy case where some value v was reported more than $t+1$ times, L gathers many statements of membership in the complementary sets $\overline{I_j}$ for every j: just as many overlapping negative *joint* bloodtests. The statements are hopefully all sufficiently numerous: $|\overline{I_j}| \geq t+1$ for every j to be gathered using threshold signatures, and yield a $PoE(v)$ for *all* values v by Lemma 16. But one could consider the bad case where, e.g. malicious players that reported set membership in the first round may not talk anymore in the second, to the point that some $|\overline{I_j}| \geq t+1$ condition does not hold anymore. Here, the key *Claim* made in the final step of the protocol, and proven in Proposition 20 is that, when synchrony holds, then if enough players escaped from some "fence" $|\overline{I_j}|$, then $t+1$ must actually have been caught by some singleton "fencepost" value $[a_j]$ in the first round, yielding a PoE(a_j).

1 Report Every player P reports his input v_P (so: either valid, or 0) to L signed with τ_P.

2 Early termination *or* Select and Request *If* L receives $t+1$ reports for the same value v, then he AGGREGATES those reports. This constitutes a PoE(v) by Lemma 16 (ii).

(α) *either* v is valid, in which case he outputs v and the PoE(v);

(β) *or* v is zero, in which case the PoE(v) constitutes a PoE(any) by Lemma 16(iii). Thus as soon as L knows a valid value (possibly from receiving it from the players, which happens if at least $t+1$ honest players have a valid input), then, let v_L be such a value, he outputs v_L and the PoE(any).

Else as soon as L knows a valid value (possibly from receiving it from the players, which happens if at least $t+1$ honest players have a valid input), then, let v_L be such a value. He select(v_L) and continues as follows. Let v_{rep} be the highest reported input. L partitions the interval $[0,\infty[$ into (at most 5) nonempty consecutive distinct intervals $I_j = [a_j, b_j]$ such that: the number $|I_j|$ of values reported in I_j, with repetitions, is strictly smaller than $t+1$. [2] More details on the construction are given in §5.2.1.

3 Testify Upon receiving the intervals, each P sends to L his input v_P (possibly again) signed with τ_P, plus for every I_j to which his value *does not* belong, a testify signed with τ_P: « my input is in the complementary $\overline{I_j}$ ». (For instance if P has input 0, then he testifies for every interval I_j not containing zero.)

4 Output *Claim*: if synchrony holds from the beginning, and if L did not already terminate in **2**, then it must hold that: for every j, there exists more than $t+1$ declarations of being in the *complementary* $\overline{I_j}$.

Assembling each of these groups of declarations with τ, the prover then obtains in particular a PoE(any) by the (i) of Lemma 16. [Indeed for every $v' \in [1, v_{max}]$, he has $t+1$ declarations *not* to have input in the interval containing v'.] He then outputs v_L along with this PoE(*any*).

The Claim in **4** directly implies Theorem 19. Indeed *either* L terminates in **2**, in which case he automatically outputs a PoE for the value he selects. *Or*, he thus outputs in 4 a PoE (any), which constitutes in particular a PoE for the value that he selected.

Now the Claim remains to be proven. The Claim follows directly from the following Proposition. Indeed the Claim states that if alternative 1 of the Proposition does not hold, then alternative 2. So only the Proposition remains to be proven.

Proposition 20. *If GST holds from the beginning, then we have the following mutually exclusive alternatives:*

1. Either *there exists a value a_j such that L received more than $t+1$* reports *from different players to have input a_j, so that he early* selects *and* outputs *in "Select and Request".*
2. Or *it holds that for all j, L receives more than $t+1$* testifies *from different players to be in the complementary $\overline{I_j}$.*

Proof. Let us assume that the first alternative does not hold: L *did not* collected $t+1$ distinct reports for any value v. So that he *did* proceed with constructing intervals I_j and requesting for testifies. Let us assume by contradiction that the second alternative does not hold either: *there exists* a j such that in the testify round, less than t players testified to be in $\overline{I_j}$.

Let M be the set of players that L does not hear of in the report round: the "missing" ones, and m their number. So that L receives $3t+1-m$ reports at the end of the report round. If synchrony holds, then all players in M must be malicious. Note $|I_j|$ the number of values received by L and reported to be in I_j (of course, players did not reported explicitly to be in I_j, since the intervals were not constructed before the request round). And likewise $|\overline{I_j}|$ the number of values reported to be in the complementary $\overline{I_j}$. Thus for every j, we have

$$|\overline{I_j}| = 3t + 1 - m - |I_j| \tag{3}$$

Let us note D_j the set of *departures*, that is, the set of players from whom L heard reports to be in $\overline{I_j}$, but from whom L *does not* hear of anymore in the testify round. Note d_j their number. Then, the asumption is thus that

$$d_j \geq |\overline{I_j}| - t = 2t + 1 - m - |I_j| \tag{4}$$

Now, observe that under synchrony, all players in D_j are malicious, and that by definition M and D_j are disjunct, so that

$$t \geq d_j + m \tag{5}$$

Combining with the previous inequality, we get

$$t \geq d_j + m \geq 3t + 1 - |I_j| - t = 2t + 1 - |I_j| \tag{6}$$

[2] Notice that he is always able to do so, because otherwise it must be the case that some value v is repeated $t+1$ times or more, in which case he should have already terminated in **2**.

and thus

$$|I_j| \geq t+1 \tag{7}$$

which contradicts the rule $|I_j| < t+1$. □

5.2.1 Details on the construction (and why the number 5)

The following explicit construction ensures our claim about the log dependency in v_{max}. Let $j=1$: $b_1 := \infty$ and a_1 be the smallest integer such that the number $|I_1|$ of values reported in $I_j := [a_1, b_1[$ is strictly smaller than $t+1$. In particular $a_1 \leq v_{max}$: otherwise this would mean that the number of dishonest reports is $t+1$ or more. Then, repeat for all $j \geq 2$ while $b_j := a_{j-1} - 1 \geq 0$: let a_j be the smallest positive integer such that the number $|I_j|$ of values reported in $I_j := [a_j, b_j]$ is strictly smaller than $t+1$.

Let us illustrate a configuration of reported values such that it is necessary to cut it into 5 consecutive intervals. Assume $t = 2t' \geq 4$, and that the prover received $3t+1$ consecutive reported values (with repetitions), organized as follows. $\left|I_1\right| = t'$, $I_2 = [a_2]$ repeated $t'+1$ times,$\left|I_3\right| = t'$, $I_4 = [a_4]$ repeated $t'+1$ times and $\left|I_5\right| = t-1$.

6 A PnS with size $\left|TSS\right|$, produced with $O(n\phi)$ communication

As in §4.2, we make here a weakened and more intrinsic Definition of PnS than Equation (1). It will be more convenient for elementary constructions, and still, it exactly addresses the property needed in the proof of Lemma 10 (ii). Precisely, this Lemma (ii) it is the only place where PnS is needed in the proof of Main Theorem 7, where the property of a $\mathsf{PnS}(\phi_{max}, \phi)$ used, is namely that it guarantees that no set of $t+1$ honest players could possibly have seen, up to being in phase ϕ, a lock certificate in a higher phase than ϕ_{max}.

Definition 21 (PnS)**.** For every honest player in phase ϕ , denote $\phi_i \leq \phi$ the highest phase number up to ϕ for which he saw a valid lock certificate lc_i. Then a *Proof of non Supermajority*: $PnS(\phi_{max}, \phi)$ for some valid $(\phi_{max}, \mathsf{lc}_{max})$ is the data of: $\phi_{max} \leq \phi$, along with some data proving that no $t+1$ honest players have their ϕ_i strictly higher than ϕ_{max}.

Relatively to such a specific set of inputs (ϕ_i, lc_i): a PnS protocol is one in which honest players in phase ϕ send one message to a designated prover L_ϕ among them, such that, upon receiving $2t+1$ well-formed messages, a (honest) prover is able to output: a phase number $\phi_{max} \leq \phi$, a lock certificate lc_{max} relative to this phase, and a $\mathsf{PnS}(\phi_{max}, \phi)$.

We still consider a $2t+1$-TSS σ. We denote L_ϕ the prover associated to phase ϕ.

1 Every player P_i sends to L_ϕ a report (ϕ_i, ϕ) (along with a lock certificate in ϕ_i if $1 \leq \phi$) along with, for *each* integer value $\phi' \in [\phi_i, \ldots, \phi]$, one testimony *signed* with σ_P, of the form: "my locked phase number up to ϕ is lower or equal to ϕ' ".

2 *Upon receiving* from $2t+1$ players such well formed messages, that is: containing a lock certificate for the claimed ϕ_i, a list of testimonies which is *consistent* with the claimed ϕ_i, and such that all the messages specify "up to ϕ" with ϕ the current phase number. Then, define ϕ_{max} the *lowest* value for which there exists $2t+1$ identical testimonies: "my locked phase number up to ϕ is lower or equal to ϕ_{max} ". Leader L_ϕ then:

- extracts this ϕ_{max} *and a* lock certificate lc_{max}) relative to ϕ_{max} from one of the $2t+1$ messages received. *Claim1*: he is always able to do so;
- AGGREGATES the $2t+1$ testimonies with σ. *Claim2*: this constitutes a PnS on (ϕ_{max}, ϕ).

Proposition 22. *The previous protocol is a PnS protocol with respect to the* locked phase numbers $(\phi_{Plock}, \mathsf{lc}(\phi_{Plock}))$ *held by honest players in phase ϕ. It has overall linear communication bit complexity, precisely in $O(n\,\phi)$, and the PnS obtained consists of at most ϕ signatures. In particular, it is of* bit size independent of the number of players.

Proof. What remains to be shown are Claim2 and Claim1. Together they guarantee that the prover is indeed able to obtain (i) a locked phase number and (ii) a PnS on it (relative to phase ϕ), as expected.

Proof of Claim2: If $t+1$ honest players have a strictly higher input than ϕ_{max}, then no $t+1$ honest players can possibly testify to have a lower input, thus the leader will never receive $2t+1$ testimonies claiming so.

Proof of Claim1: By construction of ϕ_{max}, there is a player P_j in the set of the $2t+1$ ones who issued the messages, such he reported a $\phi_j \leq \phi_{max}$ but not $\phi_j < \phi_{max}$. Therefore, this player reported exactly $\phi_j = \phi_{max}$. Thus, by well formedness of the message, his report must have been appended by a lock certificate asociated to ϕ_{max} (unless $\phi_{max} = 0$). ☐

7 Other contributions

7.1 Leaderless asynchronous Byzantine agreement

Recall the structure of the VABA of [AMS19]. Players run n executions in parallel of the first phase of the consensus of [YMR$^+$19] (with weak unanimity): one for each player playing the role of the leader. After $2t+1$ of them have at least one player which outputs, a leader election is performed to decide a posteriori which of the n leaders was the "real" one. Players erase from their memories everything related to the other $n-1$ executions. Notice at this point that in 50% of cases, this elected leader is both: honest, and in the $2t+1$ leaders who enforced an output. In which case, by construction of [YMR$^+$19], a decision certificate will be ultimately received by all players to output his honest input. In case he was not honest, in order to ensure liveness, players start n executions in parallel of the second phase of [YMR$^+$19] and repeat. *Plugging* our Theorem 7 in place of [YMR$^+$19] reduces the latency of the first phase from 7 to 5 (and of the subsequent phases from 8 to 6).

7.2 Reconciling Halting in Finite time with no Amortized Overhead, and External Validity

The modifications of [NRS$^+$20, Figure 4] are: players perform as baseline a leader-based consensus with linear complexity, e.g. the one of §2.1, possibly enriched with Strong Unanimity and a Fast Track as in §2.1. The data structure of the consensus is enriched in that players perform it simultanesouly on the actual input values (of which we denote $\ell = \log V$ the length), along with their hashes. Players refuse any certificate for a value not externally valid (see Lemma 26 for a formalization). The message complexity of Theorem 7 for this is thus $O(n\ell)$. Then: when some player outputs a value v, he generates a Reed Solomon encoding of v of degree $2t$ and length n (so with tolerance of up to t erasures). He then multicasts $H(v)$, the decision certificate for the hash $H(v)$; and sends to each player i a share $RS(v)_i$ of this Reed-Solomon encoding of v length n, along with a succinct proof that the $RS(v)_i$ is the correct encoding of the value inside the hash $H(v)$. A player i receiving such a decision certificate for a $H(v)$, along with his correctly proven share $RS(v)_i$ of v, multicasts this material (the decision certificate, $H(v)$ and $RS(v)_i$).

Proposition 23. *The overall bit communication is $n^2|H(v)||\pi| + O(n\ell)$, where $|\pi|$ is the (short) size of the proof. As soon as one honest player* outputs*, then every honest player* outputs *in two messages delay.*

Proof. The claim on the complexity is by construction.

The key additional invariant compared to [NRS$^+$20] is that any honest player which outputs v has necessarily received a (short) decision certificate for the hash $H(v)$ of the value, thanks to our double data structure. And this decision certificate itself is enough to convince any player to output v as soon as he learns the actual value of v. But by construction, the above protocol guarantees that, as soon as one honest player outputs, then every player is guaranteed to receive, in two messages delay, both: a decision certificate for $H(v)$, along with $2t+1$ distinct shares of v. These shares are enough to reconstruct the actual value of v. ☐

[A simple proof system for the above purpose is proposed in [NRS$^+$20], and consists in replacing hashes by a *Merkle tree* of the codewords $RS(v)_i$. Then, instead of the hash of v, a player who output communicates to each player i the *root* of this Merkle tree, with the codeword $RS(v)_i$, along with a proof that the codeword is indeed at place i of the Merkle tree. This simple proof has however logarithmic size in ℓ. Therefore [NRS$^+$20] consider more generic proof systems for this purpose, denoted as "cryptographic accumulators".]

8 Easy Lemmas and Various Extensions

8.1 Lemmas

8.1.1 Deriving a Fast Track from Strong Unanimity for Free

The following Lemma formalizes the trick used in 1.2.4 to compile Strong Unanimity into a Fast Track with no cost.

Lemma 24. *Consider a consensus protocol with Strong Unanimity. Then the following additional instructions enable, in addition, Optimistically fast output in two steps:*

Players initially report their inputs to the leader, signed with threshold $n = 3t + 1$. Upon receiving $n = 3t + 1$ such reports for the same *value v, the first leader* AGGREGATES *the signatures into a* fast output certificate *message for v (possibly with a multi/threshold signature mechanism), which he multicasts. Upon receiving a* fast output certificate *for a value w, a player outputs w.*

The proof for safety is that if some player fast outputs w, then he must have received a *fast output certificate* for this value w. This proves in turn that all honest players *have* input w. Thus by Strong Unanimity, no honest player can output a value different from w. Liveness follows from the fact that if all players are honest, so is the first leader. In particular, every player receives a *fast output certificate* in two messages delay.

8.1.2 Exclusivity Implies that No Other Value Can Be Unanimous

Lemma 25. *If some value v_{un} is an (unanimous) input of all the $2t + 1$ honest players in some execution of the protocol of Figures 4 + 5. Then no other value $v' \neq v_{un}$ can satisfy the Exclusivity Predicate of Definition 11. In particular no $\mathsf{PoE}(v')$ can ever be formed in the execution.*

Proof. By assumption on unanimity of v_{un}, we have that any set $\mathcal{D}$ of $2t+1$ declare messages contains more than $t + 1$ times v_{un}. So this violates the Exclusivity Predicate for v'. □

8.1.3 Enforcing External Validity

Consider as in [CKPS01, Def 4.1]: an external entity Q_{ID}, that may deliver to some honest players "certificates of validity" to their inputs.

Lemma 26. *Then all the protocols in this paper can be modified without any additional cost (not latency nor bit complexity), such that we have the following additional guarantees:*

- *in any case: the output of any honest player is* valid*;*
- *If in some phase long enough, we have a honest leader who has, in addition a certificate of validity on his input, then all honest players* output *in this phase. [This implies that we have the same liveness condition as in [CKPS01, Def 4.1] & [AMS19, Lemma 23 of full paper]: if all honest players start with a validity certificate on their input, then liveness condition of the previous protocols is unchanged.]*

The only modifications to be made are: 1) a honest player ignore propose messages sent by the leader in which the value proposed does not come with an external validity certificate, and refuses to output any value that does not come with a validity certificate. A leader always append the proposed value with a certificate, if he has any (either he received it at the beginning, of was forwarded some by some previous leader). Likewise he alwas appends the validity certicate in the lock certificates and decision certificates that he makes.

8.1.4 Default Output, when t or Less Honest Players Have an Externally Valid Input

The following construction brings the guarantee that players to output a non-valid value in finite time, if not enough of them have a valid input. This may be of use in some use-cases, e.g. [CL99]. On the other hand this may then violate the External Validity condition. For simplicity we state it explicitly on the top of the consensus of Figure 4.

Lemma 27. *The following additional specifications to the consensus of Figure 4 guarantee that all players output a value as soon as: the leader of a phase $\phi > 1$ is honest and the network fast enough.* If $\leq t$ *honest players have a* valid *input, then, the output may be Non-*valid. *However if $\geq t$ honest players have a* valid *input, then the output is necessarily* valid.

- ***0*** *a player with no valid input and who has seen no* lock certificate *so far, sends the signed $\perp$ message to the leader, along with* report$(0, \phi)$.
- ***1*** *upon receiving such $2t + 1$ $\perp$ messages, the leader* AGGREGATES *them into a threshold signature and multicasts* propose$(\perp, \phi)$ *along with a* PnS$(0, \phi)$.
- ***2*** *a player accepts* propose$(\perp, \phi)$ *if: it comes with a valid threshold signature, and, if the* PnS$(0, \phi)$ *is valid. The rest carries unchanged.*

Proof. *If $\geq t + 1$ honest players have a* valid *input*: then no set of $2t + 1$ messages can exist. On the other hand, a honest leader will always be reported of a valid value, so at least can propose it if no lock certificate was reported to him. *If $\leq t$ honest players have a* valid *input* Then a honest leader, out of a set of $2t + 1$ well-formed report messages: either he obtains a valid value, or, all these messages must then be equal to report$(0, \phi)$ and come appended with signed $\perp$ messages.

$\square$

8.2 Other Various Extensions

8.2.1 Suboptimal Adversary Thresholds, for More Tolerant Fast Tracks

The constructions we describe are illustrated on the maximal adversary threshold. They can be applied to the suboptimal threshold $n = 3t + 2c + 1$, with some parameter $c > 0$, which enables in return a Fast Track as soon as up to $2c$ players are malicious. The adaptation to the parameters of: the number of messages constituting the sets $\mathcal{R}$, $\mathcal{D}$, and the number of signatures needed for a lock certificate and decision certificate, are given in [GAG+19, DGV05, AGK+15]. The constructions of PoE and PnS (generic, or with the alternative weaker Definitions 15 & 21) carry unchanged over these new parameters.

8.2.2 State Machine Replication, (of which Possible Improvements to Libra)

In its historical conception ([CL99]), the regime of "state machine replication" may be defined as an ordered sequence of instances of consensus with Weak Unanimity, where the External Validation entity Q_{ID} is embodied by several "Clients" and the leaders denoted as "primaries". *Using* the simpler variant (§2.1) of Main Theorem 7 with plain Responsiveness and Optimal Latency in the transaction system [Lib19], in place of the consensus of [YMR+19], would result in an Optimal Latency of 6 in higher phases instead of 8 (and 5 in the first phase instead of 7).

Another limitation pointed in the preliminary version of [Lib19], is that this uses threshold signatures of size $\Omega(n)$. Indeed these signatures communicate the identities of the signers. Thus their overall complexity falls back to quadratic: $\Omega(n^2)$. This choice was apparently made to avoid a trusted setup. Instantiating Main Theorem 7 with the Transparent TSS of [ACR20], also removes this limitation, since it does not use any trusted setup.

8.2.3 Making Linear an Alternative Fast Track in 3 steps instead, preserving latency of 5 in the first phase.

Dictatorial Fast Track in 3 *messages delay.* The following Definition 1 is satisfied by [GAG$^+$19], although we define it for simplicity: in the maximal adversary threshold (see 8.2.1 for other parameters), for a single instance, and allows such a Fast Track only in the first phase. This Definition comes on the top of Definition 1 (or alternatively the Consistency and Consensus Weak Unanimity with External Validity of Definition 13) But is *not* compatible with Strong Unanimity (Definition 2 or alternatively the one with External Validity in Definition 13). It also comes at the top of Definition 4, in the strong conception where 5 messages delays are in addition required in the first phase.

Definition 1 (Dictatorial Fast Track). *We say that a consensus has an (optimistic) Dictatorial Fast Track (in* 3 *messages delay), if we have furthermore that: if all players are honest, and if the Pacemaker is forever blocked on the first phase, then, all players* output *within* 3 *messages delay from the start of the execution, and furthermore they* output *the input of the Leader.*

Under partial synchrony where messages can be lost (§3.3, [DLS88, §3.1]), this holds only if the network is initially synchronous. The reason for requiring here 5 messages delays in the first phase in Definition 4, is that the above fast delay of 3 messages circumvent the impossibility of [Ram20, B']. Notice also the Dictatorial Fast tracks in 2 messages of, e.g., [AGK$^+$15, DGV05], there denoted as "very fast learning".

Construction with Linear Complexity. The following construction uses as baseline Figure 4. We claim no originality, since it is meant to have the same structure as [GAG$^+$19], with succinct proofs used as drop-in replacement to forwarding of many messages. Players i who issued a $\mathsf{lock\,vote}(v_i, \phi = 1)$ the first phase, store the value $v_{i,fast} := v_i$ forever. The additional data structure is added: a $\mathsf{fast\,dec\,cert}(v, 1)$ for a value v (only relatively to phase $\phi = 1$) is the AGGREGATE of $3t + 1$ identical $\mathsf{lock\,vote}(v, 1)$ issued by all players.

0 Report If $\phi = 1$, skip. Else, *In addition to the* report*:* players send to L_ϕ a signed $\mathsf{declare}(v_{i,fast})$

1 Propose *If* $\phi = 1$*: unchanged.* Else: The leader L_ϕ, upon receiving a set $\mathcal{R}$ of report messages from $2t + 1$ distinct players, then, letting ϕ_{max} be the highest ϕ_j in $\mathcal{R}$:

- (i) *either* $0 < \phi_{max}$*, then as in Figure 4* ***1(i)***
- (ii) or: let $\mathcal{D}$ be the set of $2t+1$ $\mathsf{declare}$ messages received. The leader selects any value v_L reported in $\mathcal{D}$, such that *no conflicting value* $v' \neq v_L$ is reported identically in $t + 1$ messages of $\mathcal{D}$. He constructs a $\mathsf{PoE}(v_L)$ out of $\mathcal{D}$. Then he multicasts $\mathsf{propose}\big(v_L, \phi\big)$ appended with the justification $\mathsf{PoE}(v_L)$ (in addition to the justification $\mathsf{PnS}(0, \phi)$, as in Figure 4).

2 Lock vote Is enriched in case (ii): in addition to the previous check on $\mathcal{R}$, every player also checks validity of the $\mathsf{PoE}(v)$ for the value v proposed.
(Then if this checks also passes, he replies with a signed $\mathsf{lock\,vote}(v, \phi)$, as in Figure 4)

3 Lock/Fast certificate *if* $\phi = 1$*, In addition:* Leader L_1, upon receiving $3t+1$ $\mathsf{lock\,vote}(v, 1)$ for the same value v AGGREGATES them into a $\mathsf{fast\,dec\,cert}(v, 1)$, which he multicasts along with v.

4 Decision vote / Fast decision *In addition:* Players in $\phi = 1$, upon receiving a $\mathsf{fast\,dec\,cert}(v, 1)$, output v.

Fig. 9. Adding 3-steps Fast Track in $O(n\big|\mathsf{PoE}\big|)$ complexity

Improving the Throughput over Multiple Ordered Instances by "Pipelining" "Throughput", in the above "state machine replication" regime, denotes the average number of consecutive instances per phase. Throughput may be optimized with the so-called "pipelining trick" [BG17, YMR$^+$19, CS20]. Recall that this consists in having the leader at step **3** of instance i, upon forming a $\mathsf{lock\,certificate}(v_i, .)$, start simultaneously an instance $i + 1$. The $\mathsf{propose}(v_{i+1})$ of this new instance may then be appended with the $\mathsf{lock\,certificate}(v_i, .)$. Players then accept a $\mathsf{propose}$ in instance $i+1$ only if it comes appended with a $\mathsf{lock\,certificate}$. An additional variation in this regime is that leaders may rotate every step, or every two steps, instead of every phase.

Weaker Synchronizers. In weaker models, Synchronizers may designate different leaders to different players in the same phase (in particular multiple players believing themselves to be the leader). The results hold unchanged. However the Definition 6 of bit complexity would need to be changed accordingly, e.g., by dividing the bitsize by the number of honest leaders in the same phase).

9 Acknowledgements

This work was made possible thanks to early diffusion in 2018 of numerous questions and results: [AGM18, GAG$^+$19] for the question of communication complexity when enabling a Fast Track, the v1 of [CPS18] for the complexity of Strong Unanimity, and then [YMR$^+$19] v2 for the issue of Responsiveness with Optimal Latency. We are indebted to Petr Kuznetsov, including for strongly encouraging to work on this issue of Reponsiveness with Optimal Latency. We are indebted to Mark Abspoel for suggesting that a PoE for general values could be derived from the binary case, motivating the Responsive §5.1.2. We thank Antoine Durand, Sarah Bordage, Ronald Cramer, Thomas Attema and Hieu Phan for constructive discussions. We thank anonymous Reviewers for many constructive comments, and for suggesting the challenges of External Validity and of Amortized bit complexity.

References

ABC$^+$11. Jae Hyun Ahn, Dan Boneh, Jan Camenisch, Susan Hohenberger, abhi shelat, and Brent Waters. Computing on authenticated data. Cryptology ePrint Archive, Report 2011/096 (long version of TCC'11), 2011. `https://eprint.iacr.org/2011/096`.

AC20. Thomas Attema and Ronald Cramer. Compressed sigma-protocol theory and practical application to plug & play secure algorithmics. In *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 513–543. Springer, 2020.

ACD$^+$19. Ittai Abraham, T.-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 317–326. ACM, 2019.

ACD$^+$20. Ittai Abraham, T-H. Hubert Chan, Danny Dolev, Kartik Nayak, Rafael Pass, Ling Ren, and Elaine Shi. Communication complexity of byzantine agreement, revisited, 2020.

ACR20. Thomas Attema, Ronald Cramer, and Matthieu Rambaud. Compressed sigma-protocols for bilinear circuits and applications to logarithmic-sized transparent threshold signature schemes. Cryptology ePrint Archive, Report 2020/1447, 2020. `https://eprint.iacr.org/2020/1447`.

ADD$^+$19. Ittai Abraham, Srinivas Devadas, Danny Dolev, Kartik Nayak, and Ling Ren. Synchronous byzantine agreement with expected O(1) rounds, expected $o(n^2)$ communication, and optimal resilience. In *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, volume 11598 of *Lecture Notes in Computer Science*, pages 320–334. Springer, 2019.

AGK$^+$15. Pierre-Louis Aublin, Rachid Guerraoui, Nikola Knezevic, Vivien Quéma, and Marko Vukolic. The next 700 BFT protocols. *ACM Trans. Comput. Syst.*, 32(4):12:1–12:45, 2015.

AGM$^+$17. Ittai Abraham, Guy Gueta, Dahlia Malkhi, Lorenzo Alvisi, Ramakrishna Kotla, and Jean-Philippe Martin. Revisiting fast practical byzantine fault tolerance. *CoRR*, abs/1712.01367, 2017.

AGM18. Ittai Abraham, Guy Gueta, and Dahlia Malkhi. Hot-stuff the linear, optimal-resilience, one-message BFT devil. *CoRR*, abs/1803.05069, 2018.

AMN$^+$20. Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Maofan Yin. Sync hotstuff: Simple and practical synchronous state machine replication. In *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*, pages 106–118. IEEE, 2020.

AMS19. Ittai Abraham, Dahlia Malkhi, and Alexander Spiegelman. Asymptotically optimal validated asynchronous byzantine agreement. In *PODC*, pages 337–346. ACM, 2019.

ANRS20. Ittai Abraham, Kartik Nayak, Ling Ren, and Nibesh Shrestha. On the optimality of optimistic responsiveness. 2020.

BBHR18. Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. *IACR Cryptol. ePrint Arch.*, 2018:46, 2018.

BCG20a. Elette Boyle, Ran Cohen, and Aarushi Goel. Breaking the $o(\sqrt{n})$-bits barrier: Balanced byzantine agreement with polylog bits per-party. *IACR Cryptol. ePrint Arch.*, 2020:130, 2020.

BCG20b. Manuel Bravo, Gregory V. Chockler, and Alexey Gotsman. Making byzantine consensus live. In *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 23:1–23:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

Ben83. Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *Proceedings of the Second Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Montreal, Quebec, Canada, August 17-19, 1983*, pages 27–30. ACM, 1983.

BG17. Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *CoRR*, abs/1710.09437, 2017.

BIW10. Nazreen Banu, Taisuke Izumi, and Koichi Wada. Doubly-expedited one-step byzantine consensus. In *Proceedings of the 2010 IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2010, Chicago, IL, USA, June 28 - July 1 2010*, pages 373–382. IEEE Computer Society, 2010.

BKZL20. Erica Blum, Jonathan Katz, Chen-Da Liu Zhang, and Julian Loss. Asynchronous byzantine agreement with subquadratic communication. *IACR Cryptol. ePrint Arch.*, 2020:851, 2020.

BMTZ17. Christian Badertscher, Ueli Maurer, Daniel Tschudi, and Vassilis Zikas. Bitcoin as a transaction ledger: A composable treatment. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 324–356. Springer, 2017.

Bol03. Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme. In *Public Key Cryptography*, volume 2567 of *Lecture Notes in Computer Science*, pages 31–46. Springer, 2003.

CKL$^+$09. Allen Clement, Manos Kapritsos, Sangmin Lee, Yang Wang, Lorenzo Alvisi, Michael Dahlin, and Taylor Riche. Upright cluster services. In *Proceedings of the 22nd ACM Symposium on Operating Systems Principles 2009, SOSP 2009, Big Sky, Montana, USA, October 11-14, 2009*, pages 277–290. ACM, 2009.

CKPS01. Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. Secure and efficient asynchronous broadcast protocols. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 524–541. Springer, 2001.

CKS20. Shir Cohen, Idit Keidar, and Alexander Spiegelman. Brief announcement: Not a coincidence: Sub-quadratic asynchronous byzantine agreement WHP. In *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*, pages 175–177. ACM, 2020.

CL99. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, Berkeley, CA, USA, 1999. USENIX Association.

CL02. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20:398–461, 11 2002.

CM19. Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 777:155–183, 2019.

CPS18. T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. Communication-efficient byzantine agreement without erasures. *CoRR*, abs/1805.03391, 2018.

CPS19. T.-H. Hubert Chan, Rafael Pass, and Elaine Shi. Consensus through herding. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 720–749. Springer, 2019.

CS06. Bernadette Charron-Bost and André Schiper. Improving fast paxos: being optimistic with no overhead. In *12th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2006), 18-20 December, 2006, University of California, Riverside, USA*, pages 287–295. IEEE Computer Society, 2006.

CS20. Benjamin Y. Chan and Elaine Shi. Streamlet: Textbook streamlined blockchains. In *AFT '20: 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, October 21-23, 2020*, pages 1–11. ACM, 2020.

DGKR18. Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 66–98. Springer, 2018.

DGV05. Partha Dutta, Rachid Guerraoui, and Marko Vukolić. Best-case complexity of asynchronous byzantine consensus. Technical report, EPFL, 2005.

DH93. D-Z Du and F K Hwang. *Combinatorial Group Testing and Its Applications.* WORLD SCIENTIFIC, 1993.

DLS88. Cynthia Dwork, Nancy A. Lynch, and Larry J. Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, 1988.

FLP85. Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

GAG$^+$19. Guy Golan-Gueta, Ittai Abraham, Shelly Grossman, Dahlia Malkhi, Benny Pinkas, Michael K. Reiter, Dragos-Adrian Seredinschi, Orr Tamir, and Alin Tomescu. SBFT: A scalable and decentralized trust infrastructure. In *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2019, Portland, OR, USA, June 24-27, 2019*, pages 568–580. IEEE, 2019.

GV10. Rachid Guerraoui and Marko Vukolic. Refined quorum systems. *Distributed Comput.*, 23(1):1–42, 2010.

KAD$^+$09. Ramakrishna Kotla, Lorenzo Alvisi, Michael Dahlin, Allen Clement, and Edmund L. Wong. Zyzzyva: Speculative byzantine fault tolerance. *ACM Trans. Comput. Syst.*, 27(4):7:1–7:39, 2009.

KSM20. Eleftherios Kokoris-Kogias, Alexander Spiegelman, and Dahlia Malkhi. Asynchronous distributed key generation for computationally-secure randomness, consensus, and threshold signatures. In *ACM CCS 2020*, 2020.

Kur02. Klaus Kursawe. Optimistic byzantine agreement. In *21st Symposium on Reliable Distributed Systems (SRDS 2002), 13-16 October 2002, Osaka, Japan*, pages 262–267. IEEE Computer Society, 2002.

Lam98. Leslie Lamport. The part-time parliament. *ACM Trans. Comput. Syst.*, 16(2):133–169, May 1998.

Lam02. Leslie Lamport. Fast byzantine paxos, 2002. US7620680B1 patent.

Lam06. Leslie Lamport. Lower bounds for asynchronous consensus, 2006.

Lam11. Leslie Lamport. Byzantizing paxos by refinement. In *Distributed Computing - 25th International Symposium, DISC 2011, Rome, Italy, September 20-22, 2011. Proceedings*, volume 6950 of *Lecture Notes in Computer Science*, pages 211–224. Springer, 2011.

Lib19. Libra Team. *State Machine Replication in the LibraBlockchain*, 2019. Version 2019-10-24.

MA05. Jean-Philippe Martin and Lorenzo Alvisi. Fast byzantine consensus. In *2005 International Conference on Dependable Systems and Networks (DSN 2005), 28 June - 1 July 2005, Yokohama, Japan, Proceedings*, pages 402–411. IEEE Computer Society, 2005.

MCK20. Atsuki Momose, Jason Paul Cruz, and Yuichi Kaji. Hybrid-bft: Optimistically responsive synchronous consensus with optimal latency or resilience. Cryptology ePrint Archive, Report 2020/406, 2020. `https://eprint.iacr.org/2020/406`.

MR20. Atsuki Momose and Ling Ren. Optimal communication complexity of byzantine consensus under honest majority, 2020.

NBMS19. Oded Naor, Mathieu Baudet, Dahlia Malkhi, and Alexander Spiegelman. Cogsworth: Byzantine view synchronization. *CoRR*, abs/1909.05204, 2019.

NK20. Oded Naor and Idit Keidar. Expected linear round synchronization: The missing link for linear byzantine SMR. In *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*, volume 179 of *LIPIcs*, pages 26:1–26:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

NRS$^+$20. Kartik Nayak, Ling Ren, Elaine Shi, Nitin H. Vaidya, and Zhuolun Xiang. Improved extension protocols for byzantine broadcast and agreement. In *DISC*, volume 179 of *LIPIcs*, pages 28:1–28:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

PS18. Rafael Pass and Elaine Shi. Thunderella: Blockchains with optimistic instant confirmation. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2018.

Ram20. Matthieu Rambaud. The latency costs of optimistically fast output and of strong unanimity, in authenticated leader-based byzantine consensus under partial synchrony. Technical report, 2020. `https://perso.telecom-paristech.fr/rambaud/articles/nofast.pdf`.

Sho00. Victor Shoup. Practical threshold signatures. In *EUROCRYPT*, volume 1807 of *Lecture Notes in Computer Science*, pages 207–220. Springer, 2000.

SK19a. Nibesh Shrestha and Mohan Kumar. Revisiting EZBFT: A decentralized byzantine fault tolerant protocol with speculation. *CoRR*, abs/1909.03990, 2019.

SK19b. Nibesh Shrestha and Mohan Kumar. Revisiting hbft: Speculative byzantine fault tolerance with minimum cost. *CoRR*, abs/1902.08505, 2019.

TCZ$^+$20. Alin Tomescu, Robert Chen, Yiming Zheng, Ittai Abraham, Benny Pinkas, Guy Golan-Gueta, and Srinivas Devadas. Towards scalable threshold cryptosystems. In *IEEE Symposium on Security and Privacy*, pages 877–893. IEEE, 2020.

Tru20. Trustnodes.com. *StarkEx Demos Onboarding 1.3 Million on Ethereum's Blockchain For Pennies*, 2020.

YMR$^+$19. Maofan Yin, Dahlia Malkhi, Michael K. Reiter, Guy Golan-Gueta, and Ittai Abraham. Hotstuff: BFT consensus with linearity and responsiveness. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 347–356. ACM, 2019.

Zie06. Piotr Zielinski. Optimistically terminating consensus: All asynchronous consensus protocols in one framework. In *5th International Symposium on Parallel and Distributed Computing (ISPDC 2006), 6-9 July 2006, Timisoara, Romania*, pages 24–33. IEEE Computer Society, 2006.